NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

IMPLEMENTATION OF SECURE 6LOWPAN COMMUNICATIONS FOR TACTICAL WIRELESS SENSOR NETWORKS

by

David W. Courtney

September 2016

Thesis Advisor:	Preetha Thulasiraman
Co-Advisor:	Zachary Staples

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No.0704–0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE September 2016

3. REPORT TYPE AND DATES COVERED Master's thesis

4. TITLE AND SUBTITLE
IMPLEMENTATION OF SECURE 6LOWPAN COMMUNICATIONS FOR TACTICAL WIRELESS SENSOR NETWORKS

5. FUNDING NUMBERS

W6A99

6. AUTHOR(S) David W. Courtney

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Marine Corps Systems Command and Naval Research Program

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

The ability to securely disseminate data in a timely manner is critical to military missions within a hostile environment. Tactical wireless sensor networks (WSN) consist of power-constrained devices spread throughout a region-of-interest to provide data extraction in real time. In this thesis, we develop cyber security mechanisms to be implemented on a tactical WSN using the 6LoWPAN protocol for use by the United States Marine Corps (USMC). Specifically, we develop an architectural framework for tactical WSNs by studying cyber security gaps and vulnerabilities within the 6LoWPAN security sublayer, which is based on the IEEE 802.15.4 standard. We develop a key management scheme and a centralized routing mechanism that is non-broadcast but feasible in an operational scenario. In addition, we modify the 6LoWPAN enabled IEEE 802.15.4 frame structure to facilitate the newly developed keying and centralized routing mechanisms. Methods to aid in deployment planning are also discussed. The tactical WSN architecture was tested against a variety of well-known network attacks. The attacks simulated were spoofing, man-in-the-middle, and denial-of-service. Through MATLAB simulations, we showed the effectiveness and efficiency of the developed cyber security mechanisms to provide integrity and reliability to a deployed tactical WSN.

14. SUBJECT TERMS
Wireless Sensor Networks (WSN), Sensor Nodes, 6LoWPAN, IEEE 802.15.4, energy constrained node, centralized routing, keying mechanism, network attacks, denial of service (DOS), man-in-the-middle (MITM), spoofing

15. NUMBER OF PAGES 255

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

IMPLEMENTATION OF SECURE 6LOWPAN COMMUNICATIONS FOR TACTICAL WIRELESS SENSOR NETWORKS

David W. Courtney
Lieutenant, United States Navy
B.S., United States Naval Academy, 2007

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2016

Approved by: Preetha Thulasiraman, Ph.D.
Thesis Advisor

Zachary Staples
Co-Advisor

R. Clark Robertson, Ph.D.
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The ability to securely disseminate data in a timely manner is critical to military missions within a hostile environment. Tactical wireless sensor networks (WSN) consist of power-constrained devices spread throughout a region-of-interest to provide data extraction in real time. In this thesis, we develop cyber security mechanisms to be implemented on a tactical WSN using the 6LoWPAN protocol for use by the United States Marine Corps (USMC). Specifically, we develop an architectural framework for tactical WSNs by studying cyber security gaps and vulnerabilities within the 6LoWPAN security sublayer, which is based on the IEEE 802.15.4 standard. We develop a key management scheme and a centralized routing mechanism that is non-broadcast but feasible in an operational scenario. In addition, we modify the 6LoWPAN enabled IEEE 802.15.4 frame structure to facilitate the newly developed keying and centralized routing mechanisms. Methods to aid in deployment planning are also discussed. The tactical WSN architecture was tested against a variety of well-known network attacks. The attacks simulated were spoofing, man-in-the-middle, and denial-of-service. Through MATLAB simulations, we showed the effectiveness and efficiency of the developed cyber security mechanisms to provide integrity and reliability to a deployed tactical WSN.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

6LoWPAN	IPV6 over low power wide area network
AES	Advanced Encryption Standard
AES-CCM	Advanced Encryption Standard-Counter with Cipher Block Chaining-Message Authentication Code
AH	Authenticated Header
BEB	Binary Exponential Back-off
BS	base station
COC	combat operations center
CRC	Cyclic Redundancy Check
CSMA-CA	Carrier-Sense Multiple Access—Collision Avoidance
DOD	Department of Defense
DOS	Denial-of-Service
ECC	Elliptic Curve Cryptography
ESP	Encapsulation Security Payload
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPSec	IP Security
IV	Initialization Vector
LoWPAN	low power wide personal area network
LSeND	Lightweight Secure Neighbor Discovery for Low-Power and Lossy Networks
MAGID	Magnetic Intrusion Detection
MIC	Message Integrity Code
MITM	Man-in-the-Middle
MPH	miles per hour
MS	master station
NHC	Next Header Compression
NSA	National Security Agency
RPL	Routing Protocol for Low Power and Lossy Networks

RX-TX	receive-transmit
SEND	Secure Neighbor Discovery
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USMC	United States Marine Corps
WSN	wireless sensor network

EXECUTIVE SUMMARY

The Internet of Things (IoT) embraces network connectivity of everyday, low power sensor devices for two-way data communications. The IoT offers the potential to extend connectivity to sensor devices and mobile nodes at the tactical edge of the battlefield at a low cost for the United States Marine Corps (USMC) within their wireless sensor networks (WSN). A “WSN is a group of sensor nodes that are geographically distributed to provide data gathering and monitoring of tasks and events” [1]. The implementation of tactical WSNs empowers the USMC to collect data within areas of interest.

The IEEE 802.15.4 standard is a physical and data link communication protocol for low power wireless personal area networks Since the IEEE 802.15.4 standard only defines the first two layers of the Open Systems Interconnection model, another protocol must be used to provide full networking functionality for the WSN [2]. The Internet Engineering Task Force’s (IETF) 6LoWPAN (IPv6 over Low Power Wireless Personal Area Networks) is a protocol designed to work with the IEEE 802.15.4 standard [2].

Our study into the applicability of a 6LoWPAN enabled IEEE 802.15.4 infrastructure for USMC tactical sensor networking is focused on a structured, multi-hop static WSN rather than an ad hoc deployment. The objective of this thesis is to provide an analysis of 6LoWPAN by studying the standard’s ability to 1) provide energy conservation for the low power devices that are deployed, where the sensors are static; and 2) implement and use a key management scheme that has the ability to defend against a variety of cyber attacks. In order to achieve this objective, the vulnerabilities of the cyber security mechanisms used within 6LoWPAN WSN must be evaluated.

The proposed network design incorporates the following network architecture elements: master station (MS), base station/border router (BS), and sensor nodes. The MS serves as the central node of the network providing a universal connection to external domains and accessibility and administrative privileges to the sensor nodes while located at a safe remote location away from the WSN. The BS, commonly known as a sink node, is the transitional element within the WSN that connects the 6LoWPAN/internal environment to the public/external environment.. The sensor nodes are the end elements designed to attach to multiple types of sensors and to send data to the MS. The deployment of the devices is similar to that described in [3] but with some modifications to include the creation of the intended network, connection of the devices to the MS for bootstrapping, and deployment of the devices.

The MS’s administrative control mechanisms include node control, centralized routing, and keying mechanisms. Node control from the MS removes the need to send an individual to make a modification to the BS or sensor for a new task. The centralized routing mechanism allows the MS to control the routing of the WSN and the current network status with the implementation of path indication bits. Private keys are generated using the AES-CCM 128 keying mechanism, where each node has a unique key that is only shared with the MS, and the BS has a separate keying mechanism shared with the MS to secure the data over the external domain.

The proposed 6LoWPAN enabled IEEE 802.15.4 frame structure is based on the structure defined in [4] with header compression schemes. We modified the frame structure to incorporate the above discussed cyber security mechanisms. The first modification incorporates the path indication bits within the Path/Hop Limit byte. The path indication bits identify whether a primary or secondary route is used to transmit data to the MS. The second modification is the implementation of the AES-CCM 128 keying mechanism incorporating the use of an initialization vector and the message integrity code (MIC).

The experimental setup used to perform network simulations using MATLAB allows us to mimic multiple frames transiting the network at one time. The sensors in all simulations were the Magnetic Intrusion Detector (MAGID) described in [3]. We simulated the deployment of sensors along a two-lane intersection. There were four scenarios simulated, each representing a different speed of the vehicles traversing an intersection. These speeds are 25 miles per hour (mph), 35 mph, 45 mph and 65 mph. Simulated within these environments are three different types of attacks: spoofing, man-in-the-middle (MITM), and denial-of-service (DOS), each using a different method of execution to exploit vulnerabilities within a network. Throughout the simulation, the phase transitions of the nodes and frames were documented for further analysis.

The network within the simulation program incorporated a modified carrier-sense multiple access-collision avoidance with binary exponential back-off (CSMA-CA BEB) protocol designed specifically for the nodes within this thesis. The network topology used in the simulation is shown in Figure 1, where a reference number is given to each node in the figure. Many nodes have a primary and secondary route to send traffic, and each node makes a determination of which route to use. A decision to use the secondary route by one node does not imply the next node in the path also uses the secondary route. The nodes selected for each attack remain the same for all scenarios to allow for comparisons between the different network environments. Each scenario had six different network implementations with five trials/simulations per implementation. A total of 30 trials were conducted for each scenario, resulting in a total of 120 trials for the program.

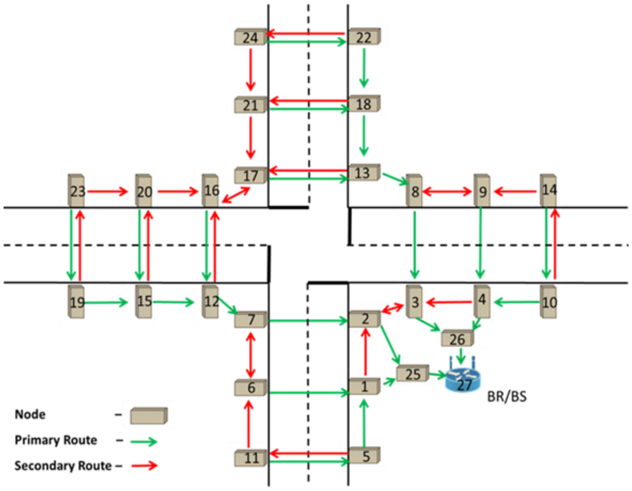


Figure 1. Assignment of Reference Numbers to the Nodes of the Tactical WSN

The purpose of the spoofing attack was to test the efficacy of the MIC security mechanism added to the IEEE 802.15.4 6LoWPAN enabled frame. The MS was successfully able to perform an analysis on frames received to determine if there was a possible spoofing attack within the WSN and to determine which node to remove from the WSN to prevent further attacks, as illustrated in Figure 2.

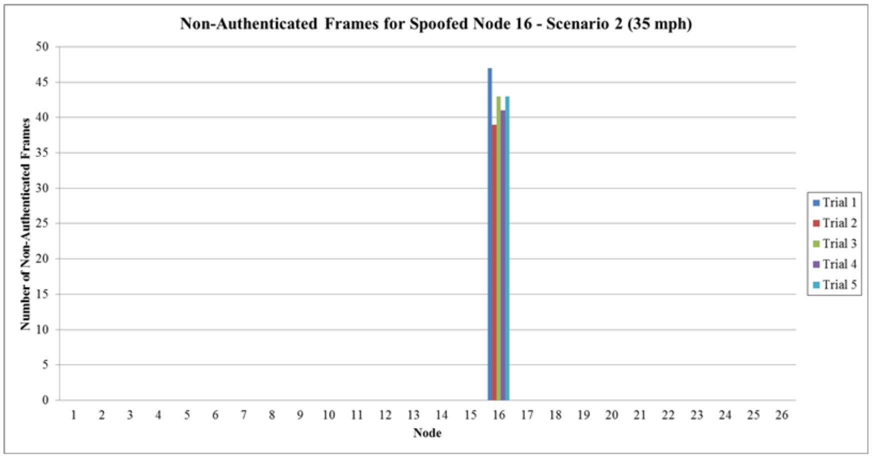


Figure 2. Number of Non-authenticated Frames Received by the MS in Each of the Five Trials for Scenario 2 Simulating a Spoofing Attack on Node 16

The DOS attack was used to determine if the centralized routing scheme and the use of the path indication bits would be able to detect an attack or incapacitated node. The analysis presented in Figure 3 displays the use of the secondary routes, indicating possible congestion or a node malfunction, causing the WSN to operate in a non-optimal manner. Specifically, during the DOS attack on node 5, node 11 had an increase of frames utilizing the secondary route due to node 5 becoming incapacitated.

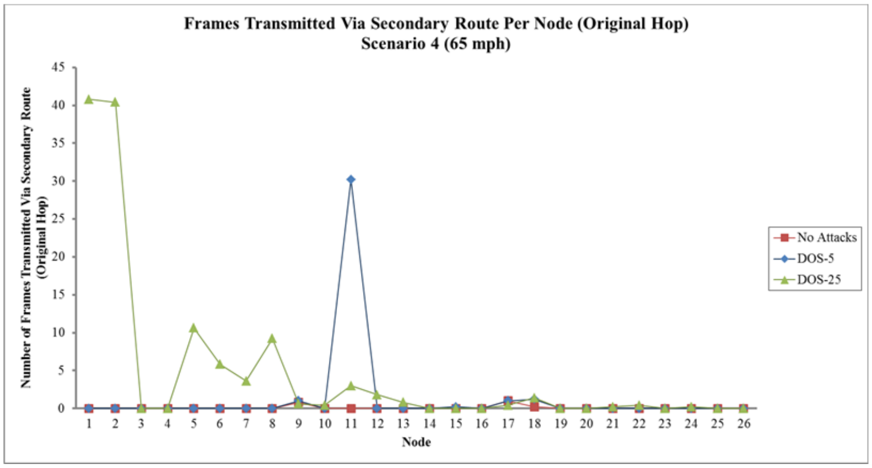


Figure 3. Frames Transmitted via Secondary Route per Node (Original Hop) in Scenario 4 (Traffic at 65 mph) for No Attacks and DOS Attacks at Nodes 5 and 25

Similar to the spoofing attack, the analysis performed on the MITM attack was focused on the implementation of the MIC security mechanism but also the centralized routing mechanism using the path indication bits. The results shown in Figure 4 validated the MIC security mechanism's ability to authenticate valid frames sent by the nodes. The MS was then able to determine where the attack was occurring and which nodes to remove from the WSN.

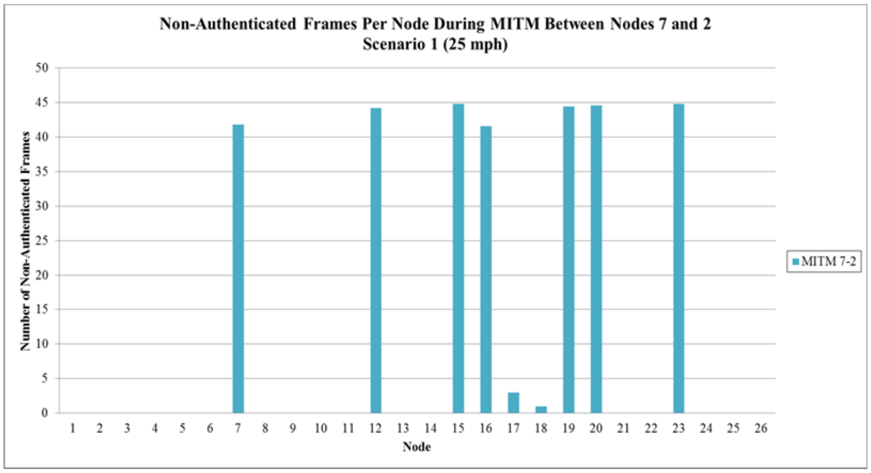


Figure 4. Number of Non-authenticated Frames Received by the MS from the Originating Node during a MITM Attack between Nodes 7 and 2

The combination of the cyber security mechanisms implemented within this thesis reflect positive results from a cyber security aspect. The use of the MIC provided integrity to the WSN by preventing the authentication of 100% of the frames received by the MS in either the spoofing or MITM attacks. The use of the centralized routing scheme ensured the WSN remained functional and reliable even when one of the two nodes connecting the BS to the rest of the WSN was disabled during the DOS attack. The implementation of the indication bits within the centralized routing scheme enabled detection of congestion within the network resulting from either traffic density or an incapacitated node. We conclude that the cyber security mechanisms developed and network structure defined provide a foundation on which future tactical WSNs used within the USMC can be based.

References

[1] K. White and P. Thulasiraman, “Energy efficient cross layer load balancing in tactical multigateway wireless sensor networks,” in *Proc. Of IEEE International Inter-Disciplinary Conference on Cognitive Method in Situation Awareness and Decision Support*, 2015, pp. 193–199.

[2] J. Granjal, E. Monteiro, and J. Sa Silva, “Security for the Internet of things: a survey of existing protocols and open research issues,” in *IEEE Communication Surveys & Tutorials*, vol. 17, pp. 1294–1312, Third Quarter, 2015.

[3] *Unattended Ground Sensor Set AN/GSQ-257 Technical Manual*, *TM 09632A-OI*, Washington, DC: U.S. Marine Corps, 2008.

[4] S. Raza, S. Duquennoy, and G. Selander, “Compression of IPsec AH and ESP Headers for Constrained Environments” (Draft), pp. 1–10, 2013, Sept. 2013.

ACKNOWLEDGMENTS

Most importantly, I would like to thank my wife, Adrienne, for all of the support she has given me throughout this entire process.

I would also like to thank my thesis advisor, Professor Preetha Thulasiraman, for taking me under her wing and pushing me to reach my goals.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The Internet of Things (IoT) embraces network connectivity of everyday, low power sensor devices for two-way data communications. The IoT offers the potential to extend connectivity to sensor devices and mobile nodes at the tactical edge of the battlefield at a low cost. In this thesis, we study the security of specific IoT protocols such that they can be used at the tactical edge by the United States Marine Corps (USMC) within their wireless sensor networks (WSN).

A. LOW POWER WIRELESS SENSOR NETWORKS

A "WSN is a group of sensor nodes that are geographically distributed to provide data gathering and monitoring of tasks and events" [1]. Sensor devices collect and transmit data to a centralized controller, also known as a base station. WSNs can be used in a variety of industrial, commercial and military applications. In recent years, WSNs have become integral to all parts of life and continue to evolve as technology advances. The use of WSNs has continuously grown and are becoming fully integrated within commercial systems as well as our homes. Due to the growth of WSNs and the increased reliability of the devices used, a WSN is an attractive intelligence gathering system to the Department of Defense (DOD). The implementation of a WSN provides the ability to remotely collect intelligence on an area-of-interest, eliminating hazardous manpower requirements. In addition, a WSN can be used to remotely "monitor deployed systems and trigger alerts at a command and control site when certain events occur" [1]. The implementation of tactical WNSs empowers the DOD to cost effectively collect reconnaissance data within critical areas-of-interest, increasing the DOD's intelligence gathering capabilities.

The deployment of tactical WSNs in remote regions requires that the sensor nodes function independently. In order to function for a period of time without manual support, the sensor devices must have their own power supply as a permanent electrical infrastructure might not be readily available. To meet the low energy consumption requirement that extends the life expectancy of the device and, thereby, the network as a whole, WSNs have been designed to use a minimal amount of power to perform a desired task.

Tactical WSNs have been used within the military during conflicts in multiple ways. During the Vietnam War, WSNs were first deployed to detect movement along the Ho Chi Minh Trail in Operation IGLOO WHITE [2]. The deployed sensors were designed to perform a range of detections to include sensing truck noises and body heat. The devices were also designed to blend in within the jungle environment by appearing to be a part of the vegetation. Operation IGLOO WHITE was determined to be a great success as WSNs, used in combination with US Air Force bombers, accounted for 90% of the destruction of equipment that was transported down the Ho Chi Minh trail.

The WSN devices in use today by the USMC for tactical networking are known as the AN/GSQ-257, Unattended Ground Sensor Set. The AN/GSQ-257 devices are part of the USMC’s Tactical Remote Sensor System and have multiple configurations that enable sensing of seismic/acoustic, magnetic, and/or infrared data [3]. This is helpful in performing perimeter enemy detection and tracking enemy movements. The use of a WSN allows the USMC to remove the human element from possible danger while maintaining situational awareness with early detection from a remote location; however, the limitations of these sensor devices include 1) physical attributes, the weight and size of these sensors are considerably higher than commercial sensor nodes, and 2) limited ability to airdrop, sensors with seismic sensing capability may be air dropped whereas all other sensing functions are only available if the sensors are hand placed. Embracing the technological advancements of sensor and sensor communication over the past 20 years will enable the USMC to modernize their network infrastructure.

B. INTRODUCTION TO 6LOWPAN/IEEE 802.15.4

One of the advantages of a sensor device is its ability to be deployed within harsh environmental conditions without a dedicated power supply. Since sensors are traditionally low power devices, a different type of communication protocol that conserves energy within the wireless environment needs to be used. The IEEE 802.15.4 standard is a physical and data link communication protocol for low power wireless personal area networks (LoWPAN). LoWPAN is commonly used in embedded applications for real-time data extraction covering a large geographic area requiring the use of many sensor nodes [4]. The sensor nodes within the LoWPAN need to be low cost as well as able to operate unattended, use typical batteries, and communicate over multiple hops [4].

Since the IEEE 802.15.4 standard only defines the first two layers of the Open Systems Interconnection model, another protocol must be used to provide full networking functionality for the WSN [5]. The Internet Engineering Task Force’s (IETF) 6LoWPAN (IPv6 over Low Power Wireless Personal Area Networks) is a protocol designed to work with the IEEE 802.15.4 standard [5]. Both IEEE 802.15.4 and 6LoWPAN are specifically tailored for IoT applications [5]. 6LoWPAN is an open standard networking technology that standardizes Internet connectivity for low power wireless sensor networks. It alters the landscape by allowing “IPv6 packets to be carried efficiently within link layer frames, such as those defined by IEEE 802.15.4” [6] while reducing Internet Protocol (IP) overhead. This low overhead is achieved using cross-layer optimizations. “A powerful feature of 6LoWPAN is that while originally conceived to support IEEE 802.15.4 low power wireless networks, it is now being adapted and used over a variety of networking media including Bluetooth and Wi-Fi” [6]. By connecting to an IP based infrastructure, low power wireless networks are then connected to other IP networks and devices via IP routers. 6LoWPAN takes advantage of the well-developed end-to-end IP infrastructure, providing open standards and interoperability [6].

C. RESEARCH MOTIVATIONS AND OBJECTIVES

As the use of WSNs grows in the USMC, they will become more attractive to potential attackers. In order to prevent a passive or active cyber attack, multiple security methods must be implemented to maintain an efficient and effective WSN. Comprehensive defense security mechanisms must account for multiple types of attacks. Generally, to defend against an attack, the military develops a defense model for the attack. Since there are multiple types of attacks, the military has developed multiple models to defend against each case. The development of a single model to defend against a variety of attacks prevents the need for an expanded arsenal of defense models, saving the military manpower.

The USMC has high interest in WSNs and their ability to connect to a public domain. Currently, when their WSN devices are deployed in the field, they are deployed with a base station, known as AN/MSC-77, which contains working spaces for two individuals [7]. The AN/MSC-77 is also known as the Combat Operations Center (COC). The COC includes a dedicated power source to provide the power necessary to run all of the equipment within it. The COC must be placed in the vicinity of the WSN devices unless a repeater is used to place the unit further away, but the COC must again retain a line-of-sight with the repeater. The size of the COC is a concern since it is a large unit, as shown in Figure 1, and can be easily seen by an enemy. The current WSN used by the USMC requires the COC to be located near the deployed network even with the use of repeaters; thus, an enemy can generally avoid the area to evade detection. The current data flow from legacy equipment and sensor devices lacks automation. In order for the USMC to obtain the data from the WSN, an individual must physically go to the COC and extract the necessary information, as the COC does not transmit the data acquired from the WSN.

To facilitate seamless data delivery to and from the sensor devices, the network must be connected to another secure domain using a comprehensive communication protocol. The use of 6LoWPAN will significantly improve the information flow as it currently exists by allowing multiple operators in a unit to access sensor information despite their location. IP based information can be easily used to inform the situational awareness and common operational picture of the engaged unit.

The feasibility of using 6LoWPAN for the operational information flow scenario described above is based on its ability to do two things: 1) provide for secure energy conserving communications with regard to the limited power devices that operate over the network and 2) provide secure communications by improving upon the security sublayer currently utilized by IEEE 802.15.4. The ability to communicate using 6LoWPAN is meaningful only if physical and cyber security considerations are in place.

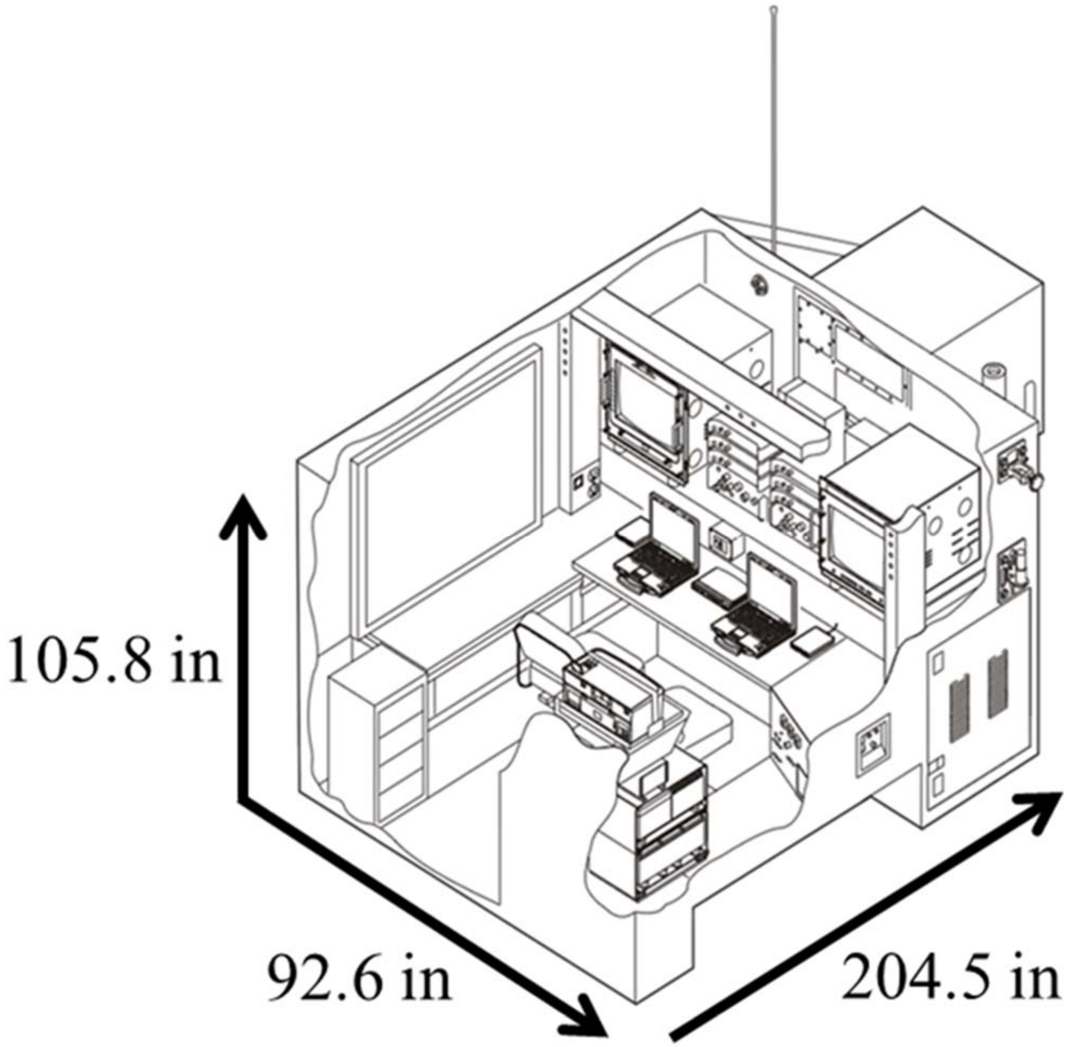


Figure 1. Dimensions of the COC (AN/MSC-77) Currently Used by the USMC. Source: [7].

In this thesis, we provide cyber security solutions for 6LoWPAN that pair with and enable current commercial sensor technology to be employed in austere and/or hostile environments to support USMC operations in a secure and energy conserving manner.

Our study into the applicability of a 6LoWPAN enabled IEEE 802.15.4 infrastructure for USMC tactical sensor networking is focused on a structured, multi-hop, static WSN rather than an ad hoc deployment. The objective of this thesis is to better understand the feasibility and effectiveness of 6LoWPAN in an operational scenario. To this end, the aim is to provide an analysis of the 6LoWPAN standard by studying the standard’s ability to 1) provide energy conservation for the low power devices that are deployed, where the sensors are static and 2) implement and use a key management scheme that has the ability to defend against a variety of cyber security attacks that could cause the WSNs to become inefficient and/or ineffective. In order to achieve this objective, the vulnerabilities of the cyber security mechanisms used within 6LoWPAN WSN is evaluated. A focus on cyber security gaps and vulnerabilities within the security sublayer, a key management scheme that is non-broadcast but also feasible within an operational scenario, is determined. Finally, the effectiveness of the selected key management scheme on different threat models is assessed.

D. THESIS CONTRIBUTIONS

To address the objectives stated above, we developed a theoretical network design framework for a multi-hop WSN that uses 6LoWPAN and IEEE 802.15.4 such that it can be deployed for tactical operations by the USMC.

The contributions of this thesis are as follows:

- Development of a command and control (administrative control) structure of the tactical WSN that incorporates node control, a unique defined/centralized routing model, and a selected keying mechanism for data confidentiality, authentication and integrity.
- Construction of a modified 6LoWPAN enabled IEEE 802.15.4 frame structure to incorporate the unique centralized routing model and selected keying mechanism.
- Enhancement of methods to aid in the deployment planning of the secured tactical WSN to prevent critical vulnerabilities.
- Simulation and evaluation of the proposed network framework against multiple attacks and testing for security robustness and energy conservation.

It must be noted that parts of this thesis have already been published by the author at the time of this writing [8].

To the best of our knowledge, this is the first work that actively studies the secure implementation of a 6LoWPAN enabled IEEE 802.15.4 WSN for USMC operations.

E. THESIS ORGANIZATION

The remainder of this thesis is organized as follows. The related literature focusing on security within 6LoWPAN is outlined in Chapter II. The theoretical framework of the proposed tactical WSN model, including modifications to the frame structure, the routing scheme and keying mechanism, is discussed in Chapter III. The basis and rationale of the experimental setup as well as the threat models tested against the developed tactical WSN are explained in Chapter IV. The simulation results and analysis are given in Chapter V. The conclusion and topics for future work are proposed in Chapter VI. All the code created and used for the implementation of the network, testing models and raw simulation results are provided within the Appendix.

F. CHAPTER SUMMARY

In this chapter, we provided an introduction and overview of WSNs, 6LoWPAN/IEEE 802.15.4 and their applicability within USMC tactical operations. The motivations and objectives of this research were discussed, followed by an outline of the thesis contributions.

THIS PAGE INTENTIONALLY LEFT BLANK

II. RELATED WORKS

There is an existing foundation of research in the literature that aims to achieve security and energy conservation within a WSN [9]. In this thesis, we focus on developing specific energy conserving security measures for a tactical WSN. To that end, we study specific tactical WSN architectures, routing mechanisms, the 6LoWPAN frame structure and related cyber security mechanisms. In the following sections, we examine current research relating to these areas to provide a basis for the work in this thesis.

A. ARCHITECTURE OF A TACTICAL WSN

The architecture determines how the network is physically deployed and how the nodes are interconnected with one another. It also determines the types of devices to be used within the network and their functions. There has been work published on tactical WSNs that serve as a foundation for our work [10], [11]. A specific view of a tactical WSN architecture was taken in [10] for the deployment of the network in a large scale environment. The authors proposed an architectural design based on a cluster-tree network with multi-hop capability [10]. The cluster-tree design groups nodes into clusters, and a node is selected to act as a cluster head. The election of the cluster head is based on a node’s residual energy level [10]. The role of the cluster head is rotated among the nodes throughout the lifespan of the deployed tactical WSN, thereby conserving nodal energy. In addition, because the network is multi-hop, it allows the network to scale to a large environment.

The architecture model developed in [10] was also used in the model developed by [11], which applied the 6LoWPAN protocol. The protocol architecture for the deployed nodes was separated into five protocol layers. Each layer was examined, detailing the functions and techniques used within the layer. The IEEE 802.15.4 standard was determined to be a suitable protocol for the physical and MAC layers [11]. While detailing the adaptation and network layers, the authors of [11] assumed that the nodes would be randomly deployed, requiring the nodes to perform a self-organizing function through address auto-configuration. Other functions that were discussed include routing protocols, header compression, fragmentation, and energy saving [11]. Within the transport and application layers, the authors used a military application management entity that contains a military operations profile specifically defining parameters required for tactical deployment [11]. While [10] and [11] provide architectural constraints for tactical WSN deployment, cyber security mechanisms and its relationship to energy consumption was not discussed.

B. ROUTING

Multiple methods of routing in a WSN exist, each of which is geared toward achieving a specific purpose such as energy conservation, low delay, or high throughput. A unique routing approach was taken in [9] where the objective was to conceal the sink node due to its high value as a target for an enemy attacker. In order to keep the sink node concealed, a routing algorithm was proposed that obfuscated the sink node's location within the deployed network, reducing the risk of attack while preserving the node's energy levels. While the algorithm provided an anonymity mechanism to conceal the sink node within the network, a cyber security mechanism for the network communications using encryption and authentication was not provided. The proposed routing mechanism within [1] focused on the development of an energy efficient cross-layer load balancing and routing algorithm called EZone. With the application of the EZone routing protocol, the network lifetime was maximized; however, the EZone algorithm does not provide a mechanism for secure communications.

In [5], the Routing Protocol for Low Power and Lossy Networks (RPL), which uses routing control messages, was proposed as a routing mechanism for networks of low power devices. "RPL is an end-to-end routing solution based on IPv6 communications and is specially adapted for the needs of specific types of traffic flow" [12] and is capable of supporting control messages on multiple network architectures [5]. One of the disadvantages of RPL is that the protocol focuses on attacks that occur externally to the network rather than internally to the network [5]. The ability to provide a method of control over the routing scheme within the network is a form of security. This type of administrative control over the routing of data is a strategic decision that is applied within this thesis.

C. 6LOWPAN FRAME STRUCTURE

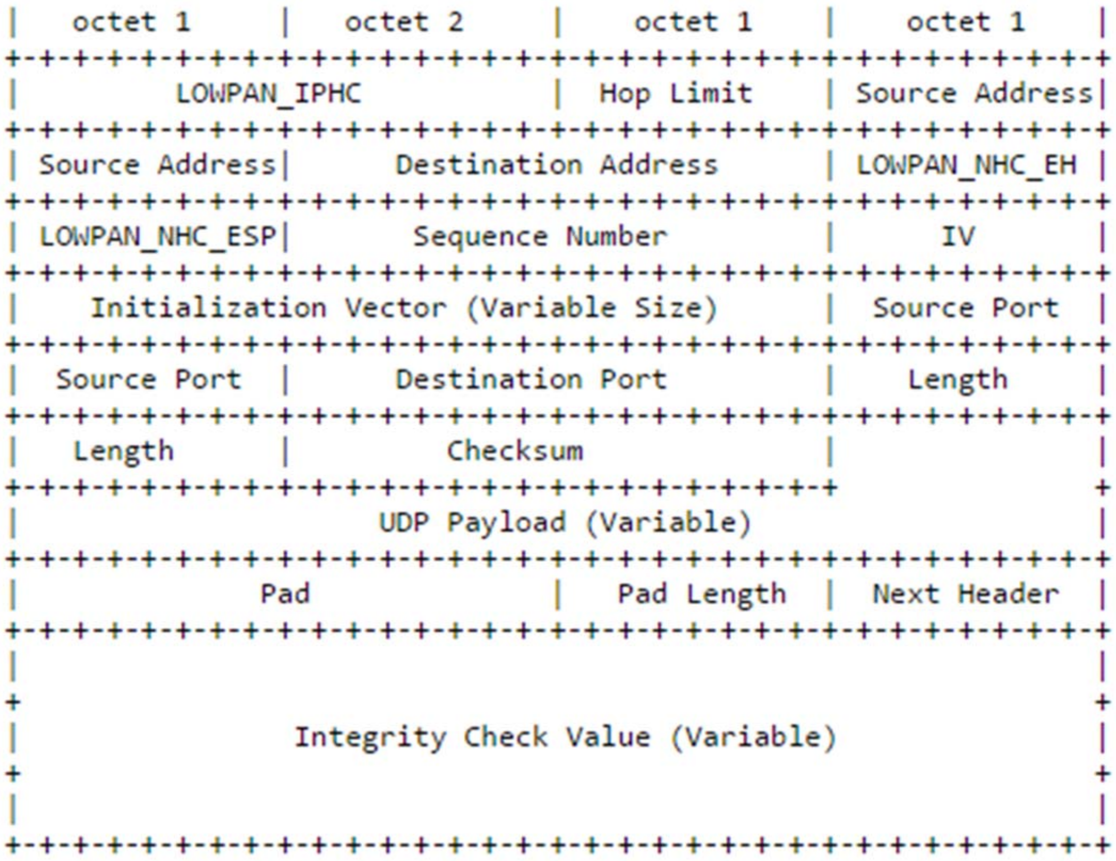
In order to develop a feasible, secure design for tactical WSNs using 6LoWPAN, it is necessary to understand its frame structure. The composition of a 6LoWPAN frame was given in [13]. Given that the packet is reduced to a size of 127 bytes, some header information has been either removed or compressed. Two of the fields within the header that underwent significant compression were the IP addresses for the source and the destination nodes. 6LoWPAN offers two types of addressing modes where the IP address is either used in its entirety or is compressed. The compressed address mode offers the administrator an option to reduce the IPv6 address from 128 bits (16 bytes) to 16 bits (2 bytes), which saves 14 bytes per address, saving a total of 28 bytes [5]. The authors of [13] also demonstrate the implementation of a Compressed IPSecurity (IPSec) packet proposed in [14], which uses a UDP packet structure within the 6LoWPAN frame. The combination of the different compression methods and the implementation of compressed IPSec within the 6LoWPAN frame is the underpinning of our proposed frame structure in this thesis.

D. SECURITY MECHANISMS

A survey on security for the IoT was conducted in [5], which went into detail on the security vulnerabilities of the IEEE 802.15.4 standard in concert with the 6LoWPAN protocol. Within the IEEE 802.15.4 standard, the authors of [5] discussed different security mechanisms depending on a prescribed vulnerability. The model proposed in [5] included the use of approved security modes, an access control list, and time synchronization to limit vulnerabilities, reducing the number of potential internal attacks; similar security mechanisms were used in [15]. The authors of [15] also provided greater detail on the potential attacks that could be used at each layer of the 6LoWPAN stack.

Encryption within the 6LoWPAN environment is a requirement in order to have an effective tactical WSN. Encryption has been addressed in the most recent release of the IEEE 802.15.4 standard [5], [15]. Multiple encryption modes are presented for use, and the challenge is for the researcher to determine what mode best fits the intended tactical WSN application. Along with encryption, there is a method to ensure data authentication called a Message Integrity Code (MIC). Again, the researcher must determine the size of the MIC needed and if the payload can sacrifice the bits needed to implement the integrity mechanism. Knowing how encryption methods operate helps determine which one to use to defend against a variety of attacks. Advanced Encryption Standard-Counter with Cipher Block Chaining-Message Authentication Code (AES-CCM) is the suggested method within the 6LoWPAN standard [5]. Within the encryption method, an Initialization Vector (IV) is used. The combined fields within the IV provide a unique value to be used along with the encryption key, creating a unique encrypted payload for each transmitted packet. The keys for encryption are either public or private. Public and private shared keys both have positive and negative aspects. The application determines the type of key to be used.

An adaptation of the 6LoWPAN enabled IEEE 802.15.4 infrastructure developed in [14] incorporates the use of encryption over the typical IP infrastructure using IPSec. IPSec is the protocol suite used for IP communications to encrypt and authenticate IP packets. The IPSec suite is an immense protocol; thus, the full implementation of IPSec is inefficient within the 6LoWPAN enabled IEEE 802.15.4 infrastructure. The authors of [14] proposed a method to reduce the overhead within the IPSec protocol while maintaining the protocol's core capabilities. IPSec on the typical IP infrastructure uses two types of headers, the Authenticated Header (AH) and the Encapsulation Security Payload (ESP) header. The AH provides authentication of who sent the packet but the data is not encrypted, whereas in the ESP header, the data is encrypted, providing confidentiality to the transmitted data. Additionally, both the AH and ESP headers can be used together within the same packet. The information fields required for the AH and ESP headers are able to be incorporated within the already used header compression for the IP address and the Next Header field. The Next Header is an 8-bit field that identifies the next type of header immediately following the IPv6 header. The 6LoWPAN packet structure including the fields of the compressed header is shown in Figure 2. The proposal by [14] further examines different types of encryption methods to be used within the ESP packet, including some of the security modes within the IEEE 802.15.4 standard.



```
|   octet 1     |   octet 2     |    octet 1    |    octet 1    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          LOWPAN_IPHC          | Hop Limit     | Source Address|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Source Address|      Destination Address      | LOWPAN_NHC_EH |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| LOWPAN_NHC_ESP|       Sequence Number         |       IV      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Initialization Vector (Variable Size)      |  Source Port  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Source Port  |      Destination Port         |     Length    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Length      |          Checksum             |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               +
|                     UDP Payload (Variable)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Pad             |  Pad Length   |  Next Header  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                Integrity Check Value (Variable)               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2. Sample Next Header Compression (NHC) Compressed IP/UDP Packet Secured with ESP. Source: [14].

The final type of security mechanism to be discussed is Neighbor Discovery. Neighbor Discovery is a method of finding neighboring nodes through which the newly added node can route messages. Neighbor Discovery is a known vulnerability of 6LoWPAN networks [5]. The vulnerability lies in verifying if the neighbor is a node that is authorized to access the WSN. Multiple methods have been proposed to deal with this vulnerability, including a Lightweight Secure Neighbor Discovery for Low-power and Lossy Networks (LSeND) addressed in [14] as well as a variety of methods presented in [5]. Methods presented in [14] include RFC 6775-”Neighbor Discovery Optimization for

6LoWPAN,” RFC 4861-”Neighbor Discovery for IPv6,” and an adaptation of RFC 3971-”SEcure Neighbor Discovery (SEND).” Given the complexity and known vulnerabilities of having a Neighbor Discovery protocol, limiting the capabilities of the network by disabling Neighbor Discovery mitigates both the complexity and vulnerabilities of the deployed WSN.

E. CHAPTER SUMMARY

In this chapter, some of the current research pertaining to the methods used for the implementation of four main elements within the tactical WSN were discussed. Also demonstrated is that an abundance of research exists on WSNs in general, but work that focuses on a deployed 6LoWPAN tactical network that is secure and uses a multi-hop infrastructure is limited. Due to the limited research, 6LoWPAN is a focal point for further exploration within tactical WSNs.

III. THEORETICAL FRAMEWORK FOR SECURITY ARCHITECTURE

In this chapter, we discuss the theoretical framework for the design and implementation of a 6LoWPAN enabled tactical WSN using IEEE 802.15.4. This theoretical framework is developed with a focus on the cyber security mechanisms that are required for tactical deployment. We discuss 1) the network design, including the network devices used and their purpose, 2) the command and control (i.e., Network administration) parameters of the WSN, which detail specific tactical characteristics within the network (i.e., data routing and key management), and 3) the type of encryption and authentication algorithm used for the transmitted data. The network design, administrative privileges and encryption and authentication of the network are the three main components of the security architecture for the tactical WSN. We then discuss the 6LoWPAN enabled IEEE 802.15.4 frame structure and the modifications that are made to the frame in order to deal with the cyber security considerations discussed. We also discuss recommendations for the deployment of the WSN for USMC tactical operations. Finally, the three types of cyber security attacks to be performed against the network are discussed. These attacks are used as an evaluation tool for the security architecture that is developed and is discussed further in Chapters IV and V.

A. NETWORK DESIGN

The proposed network design includes multiple elements, each serving a specific purpose. The elements included within the network architecture are as follows: master station (MS), base station/border router (BS), and sensor nodes. In the following sections, a description of each element is provided along with its intended use. In addition, the cyber security mechanisms associated with each network element are also provided along with assessments on attack mitigation. The proposed network architecture is shown in Figure 3 and is based on a typical mesh network.

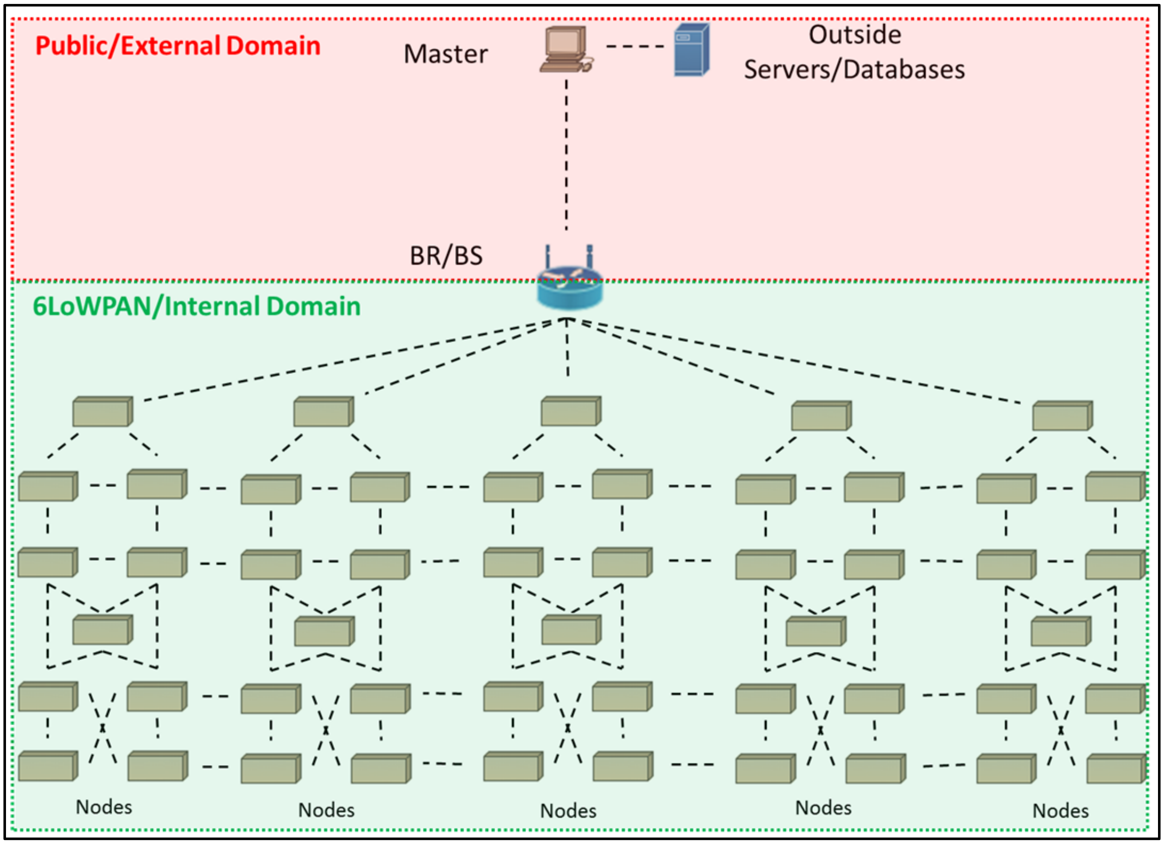


Figure 3. Proposed 6LoWPAN Network Design

1. Master Station (MS)

The MS serves as the central node of the network, as depicted in Figure 3. The proposed MS is a modified AN/MSC-77 (COC) currently used by the USMC but with modifications. As discussed in Chapter I, the typical USMC COC/MS unit must be within the line-of-sight of the WSN devices. Due to its large size, a typical COC/MS cannot be easily concealed. Rather than place the MS within the austere environment in which tactical WSNs are typically deployed, we position the MS in a structured, fortified military base, away from the tactical WSN region-of-interest, making it easier to protect. The proposed MS provides accessibility and administrative privileges to the sensor nodes while located at a safe, remote location away from the WSN. The MS has the ability to connect to each node within the internal domain since each element within the WSN supports two-way communication links. The MS can also connect to other servers outside of the WSN in order to disseminate data and populate other operator databases. When connected to other servers and databases, the data becomes accessible to remote operators, including those deployed within the WSN's environment. The MS also provides the operator a secure position to manage and control the WSN while extracting data from the sensor nodes. The flow of information proposed in this thesis does not require USMC personnel to rely on physically retrieving the stored data.

Cyber security mechanisms for the MS have already been developed and tested throughout the military (i.e., security mechanisms for the COC are well known). The MS will not be operating within the 6LoWPAN network environment but will be able to decipher the encrypted payloads sent to it from a 6LoWPAN device. As a security measure, the encrypted payloads are the only method of communication between the nodes and the MS.

2. Base Station/Border Router (BS/BR)

The BS is the transitional element within the WSN that connects the 6LoWPAN/internal environment to the public/external environment. This is also commonly known as a sink node. The proposed BS is a secured router that transmits the data received from the sensor nodes into an external domain. The BS receives frames from the sensor nodes and removes unnecessary 6LoWPAN header information. It reassembles the payload into the compatible external network frame structure in order transit to the MS. The BS performs the same task in the reverse direction, removing unnecessary frame headers and adding the appropriate 6LoWPAN header to send the frame to the sensor nodes. Within the 6LoWPAN environment, the BS converts the addresses between the internal and external network environments since 6LoWPAN uses a modified addressing mode. The BS does not interfere with the payload because it is encrypted. The BS only contains the necessary encryption in order to connect to the MS; therefore, the frame payload to be transmitted remains secure. The actual transition of frames from one domain to the other is not the focus of this thesis and is not discussed at length.

The BS is restricted to 63 hops from the furthest node since the hop limit field within the frame structure consists of only six bits. The hop-limit field is further discussed in Section D. Since the BS connects the WSN to an external network, it requires either a dedicated electrical supply, a generator, or robust battery supply as more power is needed to transmit a signal strong enough to reach the external domain. The BS is also able to withstand the harsh environments in which it is deployed and is much smaller than the COC since it does not need to have workstations available within the unit. The smaller size of the BS also allows it to be concealed more easily within the deployed environment. The BS should also contain anti-tamper technology to prevent physical modifications or reverse engineering of the device. Anti-tampering is already implemented on deployed USMC sensor devices [3]. We assume that the same tamper proof mechanisms can be implemented on the BS as well.

3. Sensor Node

The sensor nodes are the end elements. Each node is designed to attach to multiple types of sensors and to send data to the MS for compilation and analysis. The node is assumed to have the sensor capabilities as described in [3], including which sensors can be connected and which modes of operation are offered. The nodes have the ability to withstand the harsh environment and the capability to relay frames to the intended destinations. The node remains at its deployed location in order to maintain a static network. This facilitates our ability to sustain its continued connection to the MS and to monitor each node’s energy use. The sensor nodes communicate with the MS only through encrypted payload routed through the BS. The nodes are deployed and contain anti-tamper physical security measures as noted in the USMC manual [3].

4. Attack Mitigation for Network Design

Vulnerabilities associated with the design of this tactical WSN include single points-of-failure and physical protection of the sensor nodes. The MS and BS are single points-of-failure to the WSN, and if removed, the network is no longer accessible and unable to be used. A denial-of-service (DOS) attack exploits this type of vulnerability. The MS has a greater impact on the WSN since it provides reachability, accessibility, and administrative privileges to the sensor nodes. The BS can be replaced by another BS without affecting the encryption or payload data transmission between the nodes and MS.

The vulnerability of the MS and BS is not the focus of this thesis, but the military does have similar devices in place today. Lastly, the physical protection of the nodes and BS remains an accepted risk. The deployed nodes will be able to detect an enemy approaching since the sensors can detect threatening events. The ability to sense potentially hostile events allows the sensor nodes to report suspicious data before becoming compromised, and if compromised, the node is removed by the MS to prevent further corruption within the WSN. In the end, the node is still able to perform the job it was deployed to do by detecting the enemy's presence even if it becomes compromised.

B. COMMAND AND CONTROL (ADMINISTRATIVE CONTROL)

The administrative control aspects of the WSN are critically related to its functionality as well as the implementation of its cyber security mechanisms. In this thesis, the cyber security mechanisms that are implemented enable only the necessary features required for the WSN to function. These mechanisms are controlled by the MS. The control mechanisms of the MS include node control, centralized routing, and keying mechanisms. Using a centralized entity, such as the MS, to perform these functions, we limit attacks on the network.

1. Node Control

The MS maintains a directory of all of the networks connected to the BS as well as the sensor nodes within each network. The sensor nodes in the network are controlled by the MS via encrypted payload. The encrypted payload contains information that includes when the node provides real-time coverage or when the node stores detected events and transmits them in bulk at a later time [3]. Since all of the control messages are encrypted within the payload, the encryption provides the ability to securely control each node. The control of each node is limited to the MS. The MS serves as a centralized controller of all network functions. This centralization of control removes the need to send an individual to make a modification to the BS or sensor for a new task; instead the node can be adjusted immediately from the MS. In the event a node is in the process of being compromised, the MS can remove the node from the network as well as reset the node to delete its cryptographic and routing information, preventing the enemy from obtaining any data. The ability to remove the node either before, during, or after it has been compromised is a mitigating factor to the risks of a node deployed in a hostile environment.

2. Centralized Routing

As has been stated, the MS contains a directory of all nodes connected in the network. The MS implements centralized routing functions by controlling each sensor node's routing table. In the network, data is either routed upstream, from the sensor node to the MS, or downstream, from the MS to the sensor nodes. For upstream data, each sensor node is only able to route data to two neighboring sensor nodes; each node has a primary path to the MS through one neighbor and a secondary path to the MS through the second neighbor. Exceptions to this include sensor nodes that are directly connected to the BS or when a sensor node is deployed to a location in which only one other node (neighbor) is within its wireless range. The sensor node also performs in the same manner downstream, but the limitation of two nodes is not enforced in order to compensate for network expansion. This gives the MS the ability to implement energy cost saving measures within the network by adjusting the routing scheme to alleviate the workload of drained nodes. Controlling the frame flow within the network is also a way to protect the network from outside attacks by not receiving and routing frames from invalid sources.

The centralized routing scheme allows the MS to add new nodes to the network by adding the new node's address information into the neighboring node's routing table via the encrypted payload. The MS also adds the necessary routing table information to the new node during the network setup. This method prevents the need for a neighbor discovery protocol, which is noted as a vulnerability within 6LoWPAN [5], [15]. The MS can also remove compromised or expired nodes by removing the node from the neighboring nodes' routing tables; thus, any data sent from the obsolete node(s) is not routed.

An example of the centralized routing scheme of deployed sensors at an intersection is shown in Figure 4. An intersection was used as an example since these devices track not only personnel but tanks or other manned vehicles [3]. The sensors are not limited to deployment at an intersection as they may also be deployed along a perimeter of a base or along a path of intended traffic. If an anomaly within the traffic flow occurs, it may mean an impending attack or an attack by the enemy is already underway within the area. To provide full coverage of an intersection, sensors are placed on each side of the road. The primary and secondary routing paths are marked, with every node having a secondary path except for the nodes with a direct link to the BS.

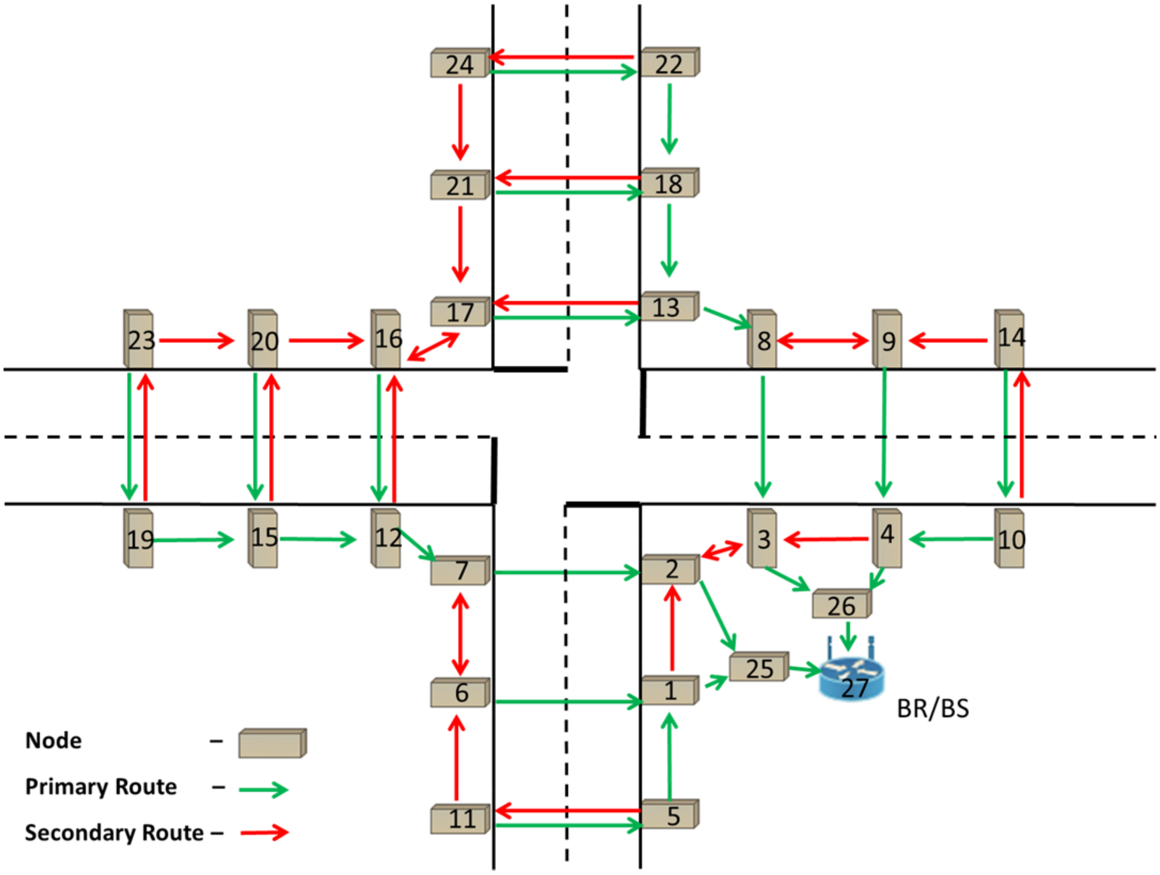


Figure 4. Centralized Routing Scheme that Depicts the Primary and Secondary Paths for Each Node

As discussed within the description of the MS, administrative control of the nodes is performed by the MS via encrypted payloads. If a node were to become compromised, the MS edits the nearby nodes' routing tables via the encrypted payloads and then removes the compromised node from the network. An example of a compromised node and the adjustments made to the primary and secondary paths of the surrounding devices is shown in Figure 5. In Figure 5, node 17 is the compromised node. The centralized routing scheme is adjusted to not allow a frame to be transmitted to or from the compromised node (node 17) as it is no longer in the routing tables of any node within the WSN. The routing adjustments depicted in Figure 5 include nodes 12, 13, 16, 18 and 21. Each of these nodes loses its secondary routes, thereby limiting its ability to transmit to only one other node. Essentially, all paths that contain node 17 are removed, thereby isolating the compromised node. The adjustments in the routing path force some nodes to assume additional traffic loads, but the WSN is able to remain effective until the compromised node is repaired or replaced.

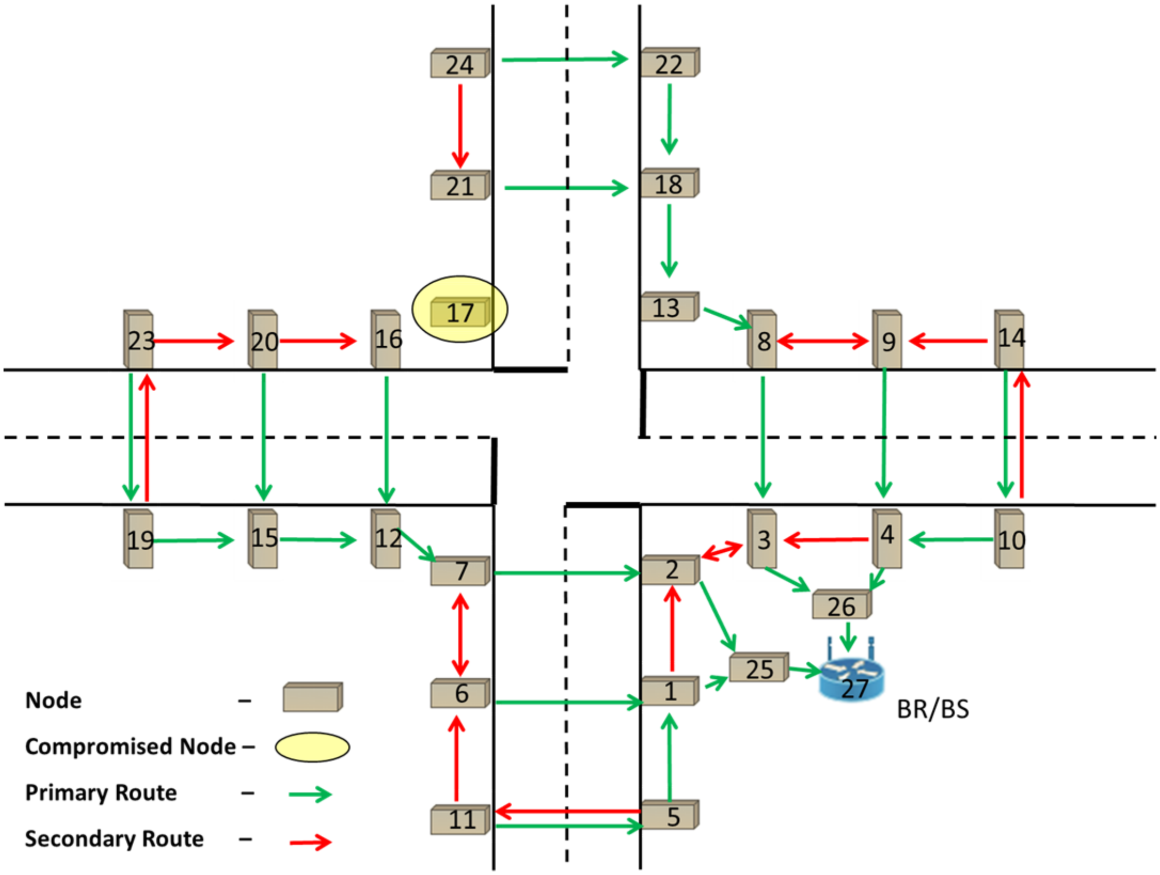


Figure 5. Centralized Routing Scheme With a Compromised Node that Depicts the Changes to the Primary and Secondary Paths of the Surrounding Devices

3. Data Transfer / Hidden Node Mitigations

A mechanism to transport data between nodes needs to be determined; however, the transporting mechanism to be used within an energy constrained environment must limit the creation of new network traffic. The authors of [14] use the User Datagram Protocol (UDP) as the basis of their packet structure, which in turn is the basis of the frame structure employed in this thesis. Unlike the Transmission Control Protocol (TCP), UDP does not use a handshake dialog for connection establishment and does not transmit acknowledgements for packets received. This allows the transfer of data to occur much faster and with less overhead [16]. UDP also supports broadcast, multi-cast, and unidirectional communications. We focus on unidirectional UDP communications as it is better suited for the keying mechanism that is developed and used in this thesis. We discuss this further in the next subsections.

As mentioned above, UDP is a connectionless, best-effort mechanism used to route network traffic. This means that the sending nodes do not receive confirmation that the transmitted frame was correctly received [16]. To provide assurance that frames are received correctly within an unreliable transport protocol (i.e., UDP), we use a unique implicit acknowledgement detection mechanism. This unique implicit detection mechanism operates as described in the following paragraphs.

As previously mentioned, the communication links between the nodes are reciprocal; therefore, each node is able to see the next-hop transmission of the frame. This confirms that the frame was properly received without errors and forwarded. If the sending node does not see the next-hop node’s transmission, the sending node determines the frame needs to be retransmitted and performs the retransmission after the predetermined back-off period. If the transmitting node is forwarding to a relay node to funnel the traffic to the BS, a small random increment of time (seed number) is added to the back-off period. The predetermined back-off period is similar to carrier-sense multiple access-collision avoidance (CSMA-CA) with binary exponential back-off (BEB) in which the time between retransmissions doubles after each failed transmission attempt [17]. The seed number is applied in order to prevent repeated attempts at transmitting a frame at the same time as a neighboring node. If the sending node does not see the transmission of the frame from the follow-on node after four re-transmissions, the sending node repeats the same process with the node designated in the routing table as the secondary route.

The primary advantage to using an implicit acknowledgment detection mechanism is the ability to ensure that a frame was received by the next-hop node without the extra network traffic required to set up sessions or send feedback control messages (acknowledgement frames) as used by other types of protocols. Essentially, this implicit acknowledgement detection mechanism takes advantage of the listening capability that each node has to detect transmissions within its transmission range. A disadvantage to this mechanism is the possibility of the originating node detecting the transmission of the follow-on node, which is transmitting a previously received frame from another node. The receiving node then uses the transmitted frame by the follow-on node as confirmation of a successful receipt.

This implicit acknowledgement detection mechanism cannot be applied to the nodes next to the BS as a result of the BS's transition and subsequent transmittal of the frame into the public domain. The neighboring nodes of the BS is not able to detect the follow-on transmission the BS makes since it is beyond the node's capabilities. With the assumption that the BS has a dedicated power supply, the BS is continuously awake and ready to receive any frame sent to it. All of the traffic within the network is then funneled into the nodes surrounding the BS, which increases the network traffic. By limiting the BS's neighboring nodes to one transmission per frame, we reduce the possibility of congestion. With the increase in traffic density of the BS's surrounding nodes, hidden nodes also become a factor to consider. A hidden node scenario is illustrated in Figure 6, where node A and C are not able to detect each other's transmission while transmitting to node B; therefore, node A is hidden to node C and vice versa. Hidden nodes within the network also create congestion because transmissions can collide, causing the received frame to be corrupted.

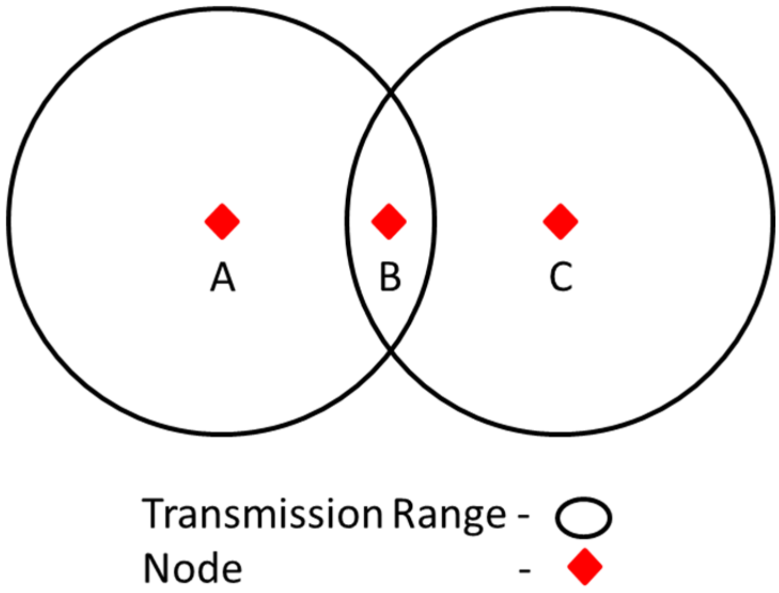


Figure 6. Hidden Node Diagram

In order to mitigate hidden nodes near the BS, extra relay nodes are used to funnel the traffic toward the BS, thereby removing possible hidden nodes from around the BS. A network implementation without the use of the extra relay nodes to funnel the network traffic is shown in Figure 7. Nodes 1 through 4 can transmit to the BS; however, nodes 1 and 2 cannot see node 4 and nodes 3 and 4 cannot see node 1, creating a hidden node problem. If nodes 1 and 4 transmitted at the same time, a collision would occur, and the frame would be lost. The use of utilizing extra relay nodes within the vicinity of the BS to avoid hidden node problems is shown in Figure 8. Within the network design and prior to deployment, the BS is moved further away from nodes 1 through 4 to allow for the addition of nodes 25 and 26 to the network. The BS is then far enough away to not be affected by nodes 1 through 4 and can only see nodes 25 and 26, while node 25 can only see 1, 2, 26 and the BS and, similarly, node 26 can only see nodes 3, 4, 25 and the BS. Nodes 25 and 26 are now functioning as a relay/sensing node and can see each other within the network. Since the extra relay nodes can see each other, the nodes do not transmit at the same time, preventing collisions from occurring at the BS. Also since nodes 25 and 26 are extra relay sensing nodes, they are able to concentrate on sensing the area around the BS to provide further physical security to the high value device.

One final modification was made to the delay in time between the first and second transmissions of a node. The first transmission is typically used to wake-up the receiving node, which affects the implementation of the BEB protocol. Traditionally, the BEB waits zero or one time slot before attempting a first transmission. Thereafter, the wait time doubles (two slots, four slots, eight slots, etc.) before each re-transmission. In our implementation, the BEB is modified as follows. A node waits 6.0 ms after the first transmission before trying to re-transmit. The transmitting node does not know if the next-hop node is awake, thus this time permits the next-hop node to wake-up in preparation to receive the incoming frame for the second retransmission. The BEB begins after the second transmission but with an initial back-off time delay of 10.0 ms which then doubles after each successive unsuccessful attempt to transmit to the follow-on node. Thus, in our implementation, back-off times between the first and second transmissions do not follow the typical BEB delay of zero or one time slots; doubling of wait time does not occur until after the second transmission.

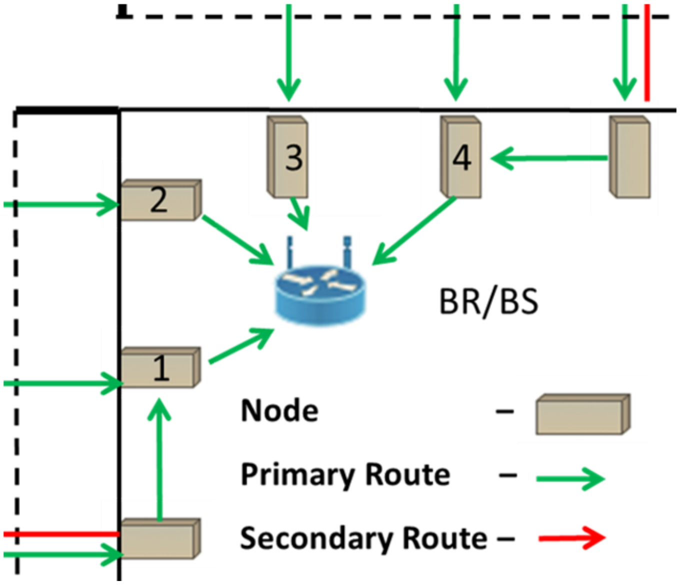


Figure 7. Network Design Without the Use of Extra Relay Nodes to Illustrate the Hidden Node Problem

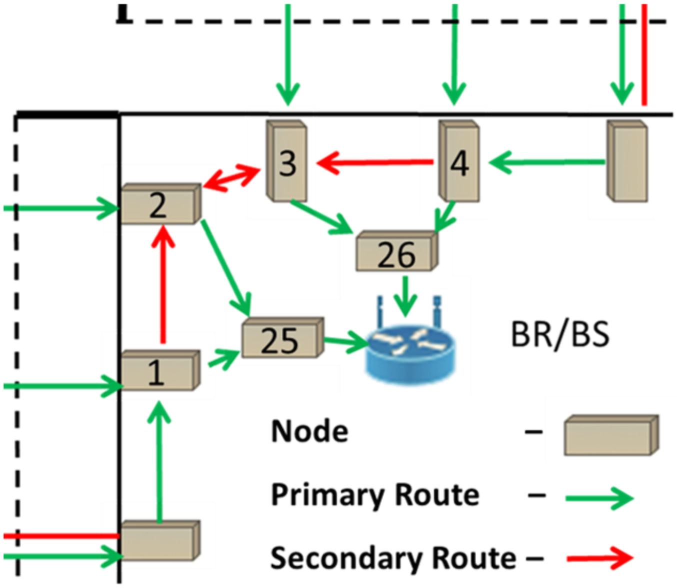


Figure 8. Network Design With the Use of Extra Relay Nodes (Nodes 25 and 26) to Illustrate a Remedy for the Hidden Node Problem

Each node has the ability to store frames that cannot be successfully forwarded. After a period of time, the node repeats the same process to forward the stored frame. By waiting a period of time without retransmitting, the node does not add to network congestion, thereby reducing possible collisions or delays. This is a capability already demonstrated within [3] in its ability to store all events and transmit them at a specified time given by the network administrator.

4. Keying Mechanisms

A keying mechanism is needed to protect the information being transported between the nodes and the MS. Multiple keying mechanisms may be implemented, such as a private keying mechanism, in which each node has its own unique key, or a public keying mechanism, in which each node has both a private key specific to each node and a public key that is shared between all of the nodes. Previous research from [5] determined that after evaluating the energy requirements of public keying, private keying is the most efficient method to encrypt the payload. Public keying was also determined to have vulnerabilities, including an increase in transmissions throughout the network to update the keys. The focus of this thesis is to evaluate a non-broadcast key management method;

therefore, individual frames are also used to disseminate each key update, creating more transmissions and, in turn, increasing energy use.

With the use of a private key, each node has a unique key that is only shared with the MS. As mentioned previously, the BS does not have any of the encryption keys shared between the nodes and the MS, but the BS does have a separate keying mechanism shared with the MS. The external network and, specifically, the keying mechanism for the external network are already in use in other areas within the military and is not the focus of this thesis.

5. Attack Mitigation for Administrative Control

Administrative control allows for mitigating factors if the WSN has nodes that are attacked via man-in-the-middle (MITM) or DOS. Each of these attacks requires an individual or remote device to be near the WSN, but the attacker is detected prior to performing the attack due to the sensing capabilities of the sensor node as previously mentioned above [3]. The centralized routing scheme prevents the intruder from further infecting and draining the rest of the network's power resources. Use of a keying mechanism also protects the data during transportation over the network; therefore, MITM or spoofing attacks are not able to change any of the data nor are they able to eavesdrop.

C. ENCRYPTION

Multiple encryption algorithms based on private keying are available. The algorithms include the Advanced Encryption Standard (AES), Elliptic Curve Cryptography (ECC), and RSA. Each encryption method is authorized by the National Security Agency (NSA), which sets the encryption standards for the Department of Defense and establishes key lengths for various classification levels [18]. The most recent IEEE 802.15.4 standard lists eight security modes ranging from no encryption (one mode) to different versions of AES (seven modes) [5]. As a result, AES is used as the keying mechanism in this thesis. The seven modes of AES boast different levels of encryption and authentication. Since the devices being used can be located within a hostile environment and are interacting with government networks, the highest levels of security are required within the WSN; therefore, the data must be encrypted and authenticated. To meet these requirements, AES-CCM with 128 bit keys is used as the keying mechanism to provide confidentiality, integrity, and authentication. It is shown that the selected encryption method also protects against MITM and spoofing. The keying mechanism is further discussed in the next section.

D. 6LOWPAN ENABLED IEEE 802.15.4 FRAME STRUCTURE

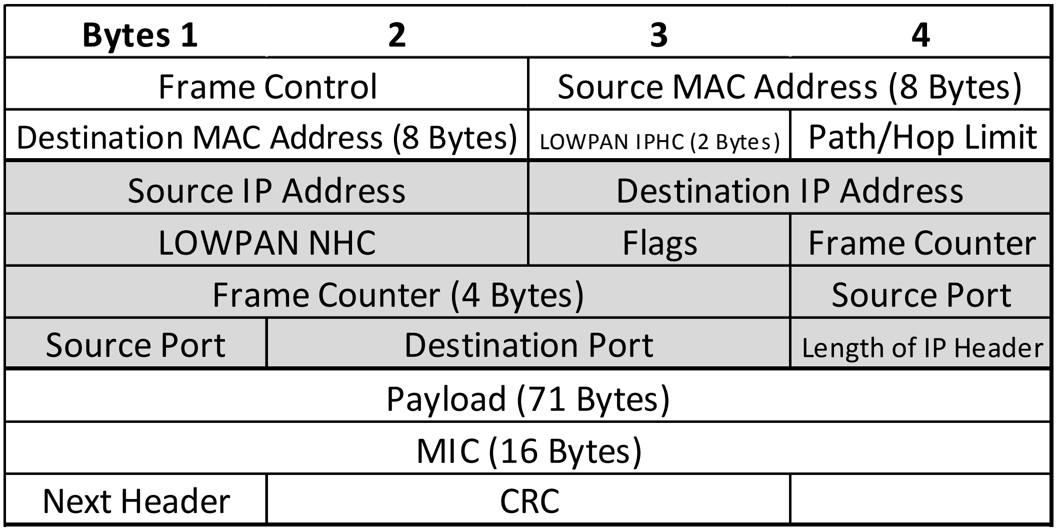
The proposed 6LoWPAN enabled IEEE 802.15.4 frame structure is shown in Figure 9 and is based on the structure defined in [14] with header compression schemes. In this thesis, we modified the frame structure to incorporate the previously discussed cyber security mechanisms. The fields within the frame structure are defined in the following subsections.

1. Frame Control (2 Bytes)

This field has been defined by the 802.15.4 standard [13].

2. Source MAC Address / Destination MAC Address (8 Bytes Each)

This field has been defined by the 802.15.4 standard [13].



<table>
<tr><th>Bytes 1</th><th>2</th><th>3</th><th>4</th></tr>
<tr><td colspan="2">Frame Control</td><td colspan="2">Source MAC Address (8 Bytes)</td></tr>
<tr><td colspan="2">Destination MAC Address (8 Bytes)</td><td>LOWPAN IPHC (2 Bytes)</td><td>Path/Hop Limit</td></tr>
<tr><td colspan="2">Source IP Address</td><td colspan="2">Destination IP Address</td></tr>
<tr><td colspan="2">LOWPAN NHC</td><td>Flags</td><td>Frame Counter</td></tr>
<tr><td colspan="3">Frame Counter (4 Bytes)</td><td>Source Port</td></tr>
<tr><td>Source Port</td><td colspan="2">Destination Port</td><td>Length of IP Header</td></tr>
<tr><td colspan="4">Payload (71 Bytes)</td></tr>
<tr><td colspan="4">MIC (16 Bytes)</td></tr>
<tr><td>Next Header</td><td colspan="2">CRC</td><td></td></tr>
</table>

Figure 9. Proposed 6LoWPAN Frame Structure

3. LOWPAN IPHC(2 Bytes)

This field has been defined by RFC 6282 [19] with no changes made.

4. Path (2 Bits) /Hop Limit (6 Bits)

The proposed centralized routing mechanism defined in this thesis limits the direction each frame can take to reach the BS from the transmitting node. This mechanism is a modification from the proposed frame structure in [14]. We use the Path/Hop Limit field in the following manner. The Path/Hop Limit field is a total of eight bits (one byte). Within the Path/Hop Limit byte, the first two bits are used to help the MS determine if there is an issue with a node routing frames. Specifically, the first two bits are used individually to determine whether the frame was transmitted over a primary or secondary route. The second bit is used only by the source node. In the event the primary route is used to send the frame, the bit is 0. If the secondary route is used, then the bit is 1. The same method is applied to the first bit and is used by all nodes except the originating node. When the MS receives the frame, it is known if a node was not able to transmit to a designated primary node. Depending on the modes of operation selected, the MS may be able to determine which node may be off line or compromised instead of waiting for a response or detection. The final six bits limit the number of hops a frame can take to 2^6–1, or 63 hops. Limiting the number of hops to 63 does not present an issue since the nodes adjacent/neighboring to the BS are not able to support a large network of nodes. In other words, we want to maintain energy efficiency within the network. By reducing the possible number of hops within the network, we provide battery relief to those nodes, particularly those near the BS, that have to transmit data regularly.

5. Initialization Vector (16 Bytes)

The IV is used to help protect against replay attacks and is also used in the CCM process to encrypt the payload [5]. The IV is shown as the shaded portion of the frame in Figure 9. To prevent replay attacks, the IV is a combination of unique identifiers. These identifiers include a frame counter (four bytes), source and destination IP address (two bytes each), source and destination port address (two bytes each), flags (one byte), and length of packet (one byte). The combination of these parameters is unique and prevents duplicate IVs; in addition, an IV cannot be repeated during the life of the network. The sequence number field has been removed from the frame structure originally given in [14].

a. Source IP Address / Destination IP Address (2 Bytes Each)

The addresses are in the compressed 16-bit mode, thus incurring smaller overhead as described within [14].

b. LOWPAN NHC(2 Bytes)

This field is defined by RFC 6282 [19] with no changes made.

c. Flags (1 Byte)

This field is reserved for the originating source and each bit is designated by the administrator of the WSN.

d. Frame Counter (4 Bytes)

The field keeps track of the sequential order of frames sent; therefore, a separate sequence number field outside of the IV is not required.

e. Source Port /Destination Port (2 Bytes Each)

These fields have no changes or compression modifications [14].

f. Length of IP Header (1 Byte)

This field is reduced to one byte since the maximum frame size is 127 bytes.

6. Payload (71 Bytes)

The payload is the amount of data that can actually be transmitted. The data is encrypted, providing confidentiality during data transmission using the combination of the IV and the AES-CCM 128 bit key.

7. Message Integrity Code (16 Bytes)

To provide authentication and integrity, a MIC is created within the AES-CCM mode of encryption and is attached to the end of the frame. The MIC is a hash unique to the packet and is used to verify that no changes were made to the original message. The MIC provides another layer of protection against any attack that tries to inject or change data being transmitted.

8. Next Header (1 Byte)

This is used in higher layers and remains unchanged [14].

9. CRC (2 Bytes Each)

This field has no changes or compression modifications [14].

E. TRANSITION

As previously mentioned, the transition of the frame from one domain to the other must be completed by the BS. Since the 6LoWPAN frame contains information that the MS needs to properly assess the effectiveness and efficiency of the WSN, the BS must transfer information from the necessary fields of the 6LoWPAN frame to the frame used on the external domain. Fields needed by the MS include the Path, IV, Payload, Next Header, and MIC. Not included in the IV but needed by the MS to analyze the route taken within the WSN are the path indicator bits; however, this field is only required when the BS is transitioning a frame from the WSN to the MS. The path indicator bits are not required when making the transition from the MS to the WSN since the nodes deployed within the WSN do not have the capability to perform that type of analysis.

F. DEPLOYMENT OF NODES

The deployment of the WSN is similar to [3] but with some modifications. It is proposed to first create the intended network, then connect the devices to the MS, and finally deploy the devices. Creating the network first requires setup of the key exchange for encryption purposes, the routing table to be loaded, and evaluation of ideal physical placements for the nodes. The administrator determines the physical location of the node within the designated network.

1. Key Exchange/Routing Table

After the network is designed and all of the routing tables have been constructed, the information for each node needs to be transferred to the node and BS. Each node and BS is physically connected to the MS for bootstrapping. The private key for the device and the constructed routing table is transferred to the node and BS. The routing table is transferred to the nodes by the MS to enable the nodes to connect to the network. The key exchange consists of a private key that is only shared between the MS and the node. The physical transfer of each unique key exists to prevent an enemy from gaining access to an entire network's information simply by obtaining the key from one node. By using a private key unique to each node, the enemy only has access to that node's information. This also allows the MS to remove the node from the network by adjusting routing tables of surrounding nodes without compromising the rest of the network. Using a key also allows the MS to perform other security procedures, including resetting the node completely before it is compromised, thus preventing the enemy from obtaining information stored on the node.

2. Physical Placement

While connected, the MS is able to maintain a geographical map of deployed nodes and map the deployment of any new node. This helps determine if the pending placement of the new node is able to connect to surrounding nodes. This is critical to the deployment of the WSN since it helps track enemy movements and position. Since the MS is also able to compile data from all deployed nodes, this allows the data to be utilized by more entities. For example, warning messages can be automatically disseminated to surrounding units if a certain threshold is met, alerting them of enemy activity within the area.

3. Network Connection

Since the physical location of the node is known and the surrounding nodes within reach, the placement of the nodes within the network can be determined. The MS can then add the node to the desired network. If the node can be added to multiple networks, then the MS must determine which network is the most energy efficient network to add the node.

G. PROPOSED ATTACKS

Three different types of attacks are conducted in the simulated environment: spoofing, MITM, and DOS. Each attack uses a different method of execution and exploits different vulnerabilities within a network, but as previously reviewed, the implemented cyber security mechanism can either prevent or lead to the detection of an attack, which triggers the deployment of mitigation methods that maintain the integrity of the network. The attacker in each of the following scenarios is assumed to be an experienced hacker. Under this assumption, the attacker knows what MAC and IP addresses to use within the injection frame as well as the key used for the cyclic redundancy check (CRC); however, it is assumed that knowledge of the pre-shared key between the node and the MS is not known.

1. Spoofing

Spoofing is simulated by a rogue node pretending to be a legitimate node within the network. To conduct the attack, a device is required to be within range of the next-hop node. The frame is then transmitted to the next-hop node to increase the probability that it will traverse the network. Operating under the assumption that the attacker is experienced, the frame successfully reaches the MS. The MS then examines the frame and determines the frame is not authentic since the MIC does not match, therefore dropping the frame. The MS also records the node the frame was from for future comparison. After multiple frames have been received, the MS can analyze the results and determine if there is a rogue node within the network. The actions of the rogue node can be mitigated by removing the node that was initially spoofed, denying all traffic sent by either of the nodes.

2. MITM

A MITM attack is simulated by placing a node between two network nodes. When a network node transmits a frame, the attacking node intercepts and changes the data within the frame before transmitting the original frame to the next-hop node. The frame still transits the network and reaches the MS. The MS examines the frame and determines it is not authentic while recording the result within its logs. Since a MITM attack affects more than one node, it is assumed that the MS can rapidly determine that a rogue node has invaded the network. The neighboring nodes affected by the attacking node can then be removed from the network.

3. DOS

A DOS attack is the disruption of services to a specific node within the network. To simulate a DOS attack on a node, a continuously transmitting rogue node is placed near a network node. This keeps the affected node constantly receiving from the rogue node, which prevents the network node from transmitting any detected events or forwarding any frames from other nodes. Using the path indicator bits within the frame, the MS can determine where the disruption is occurring and modify the networks routing tables to mitigate the disabled node.

H. CHAPTER SUMMARY

Within this chapter technical, physical and administrative controls were discussed in order to provide a secure networking environment for a deployed tactical WSN. The physical infrastructure of the network design was discussed along with technical controls for each element within the design. To provide administrative controls over the deployed tactical WSN, a centralized routing mechanism is used in conjunction with a route indicator (path bits). A modified implicit detection mechanism is also implemented to provide assurance of transmitted frames between nodes. Next a keying mechanism was selected to provide confidentiality, integrity, and authentication to the information sent. In order to implement the proposed network design, centralized routing mechanism, and keying mechanism, the frame structure was modified and discussed. Finally, in order to deploy a secure, tactical WSN, the preparation of the tactical WSN was discussed and critical steps were recommended.

IV. EXPERIMENTAL SETUP

In this chapter, we discuss the experimental setup used to perform network simulations using MATLAB. These simulations test the efficacy of the cyber security mechanisms implemented in the theoretical framework. We start by describing the characteristics of the sensor nodes used in the simulations, the phases of operation of each sensor node and the frame parameters. We then describe the network parameters in relation to the simulation. Lastly, we discuss the simulation program including the different WSN modules that were developed, the required user files, and the simulation logs.

A. SENSOR PARAMETERS

The sensor-node attachment for the purpose of detecting events is not the focus of this thesis; however, some detection parameters are assumed in order to have a fully functioning network simulation. The sensors in all simulations are the Magnetic Intrusion Detector (MAGID) described within [3]. This sensor is designed to detect large vehicles such as tanks and small vehicles as well as individuals; although, detection ranges vary depending on the power level being used and the element size.

During the simulations, sensors are set on a low power setting and emplaced along a two-lane intersection. The deployment of the nodes with an attached MAGID are shown in Figure 10. The low power setting has a maximum range of 15.0 m for the detection of vehicles, whereas the maximum range for the detection of individuals is 4.0 m [3]. The ranges are illustrated by the ellipses in Figure 10 to exhibit the detection areas of the deployed sensors. The extra relay nodes near the BS are not shown in Figure 10 since the focus of the figure is to illustrate sensor coverage.

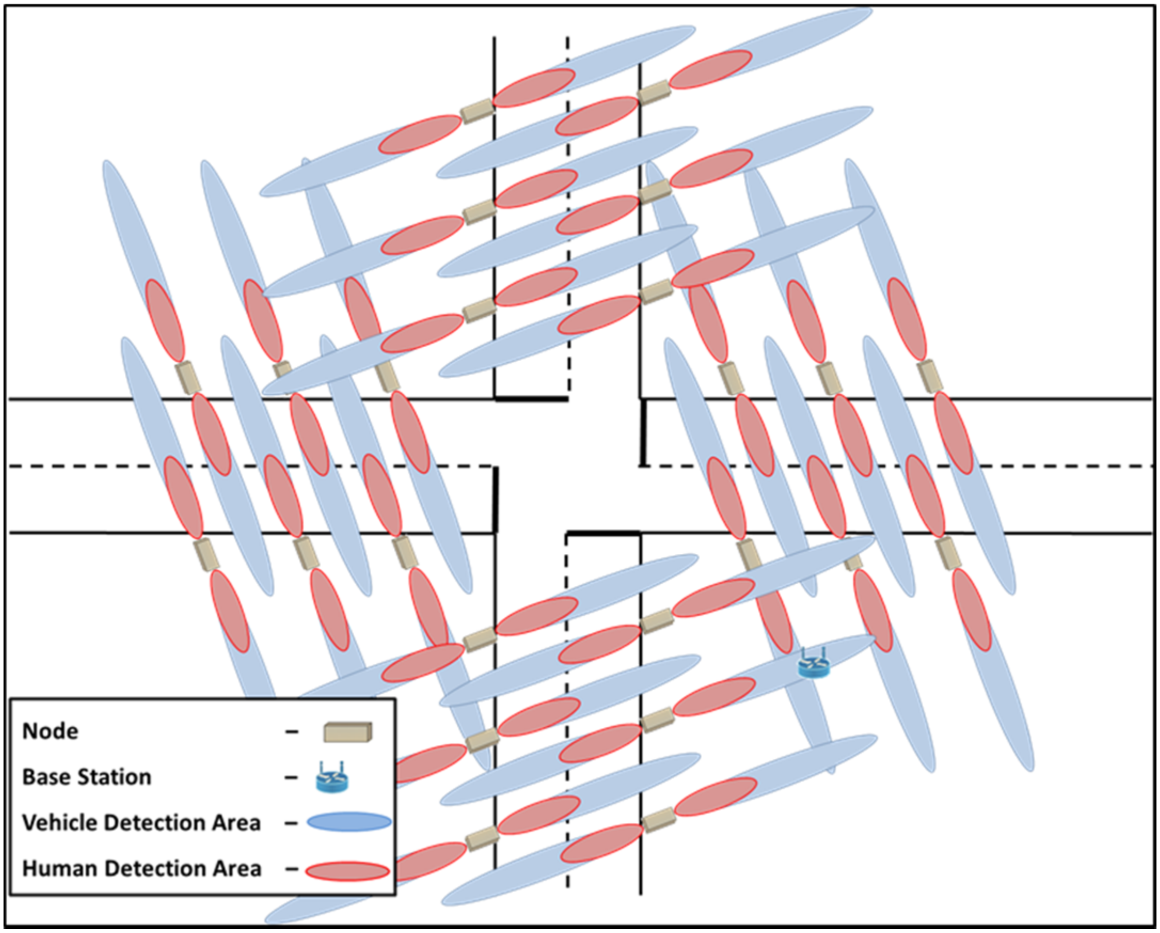


Figure 10. MADIG Deployment Coverage of an Intersection (Extra Relay Nodes Providing Coverage to the BS are Not Displayed)

As mentioned in Chapter III, the extra relay nodes are used to funnel the network traffic to the BS and prevent hidden nodes. In addition, the extra relay nodes provide coverage to the WSN's only single point of failure, the BS. A close up view of the BS, demonstrating how the extra relay nodes function together as a sensing node and the resulting coverage is shown in Figure 11.

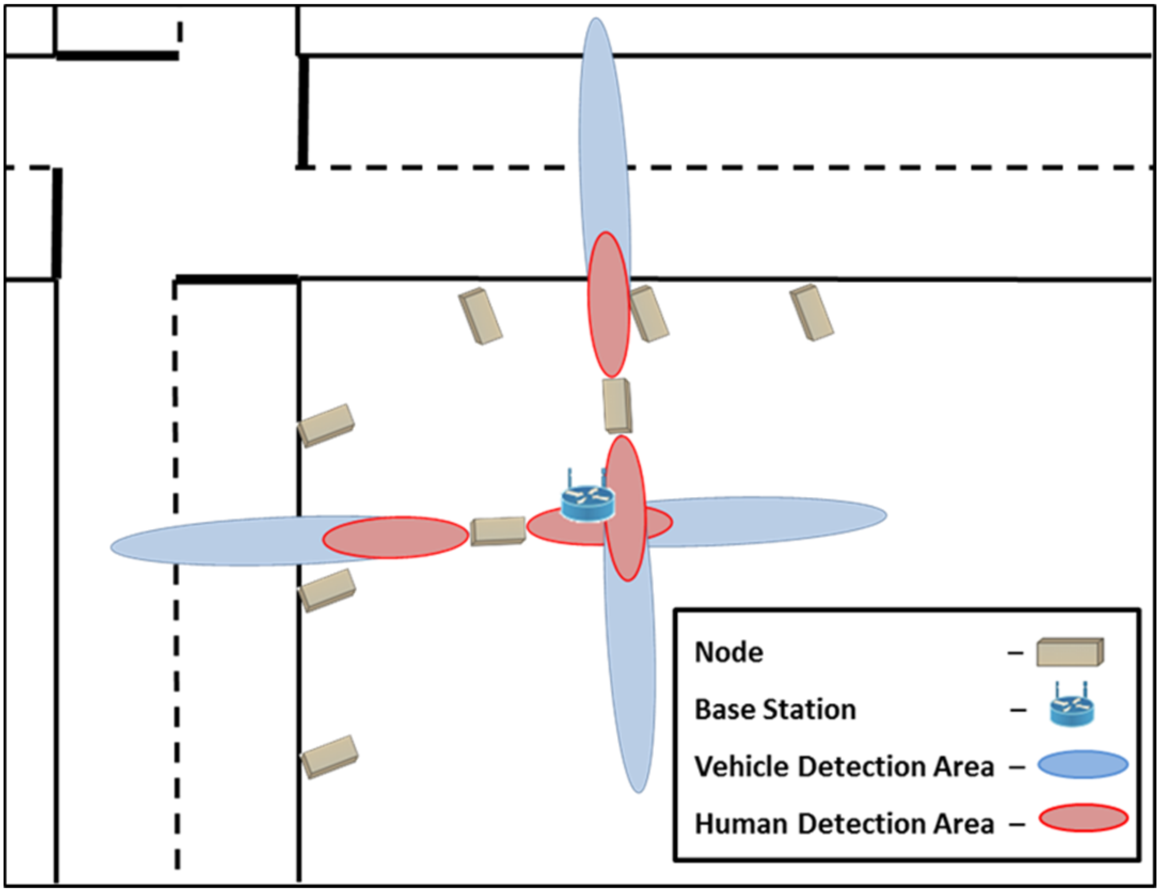


Figure 11. Close-up View of the MAGID Deployment Coverage of the Extra Relay Nodes Used to Funnel Traffic to the BS and Avoid Hidden Nodes

B. NODE CHARACTERISTICS

Each node transitions between multiple phases of operation/execution. The various phases of node operation along with the amount of power (in mW) that is used at each phase and the duration of each phase (in ms) are outlined in Table 1. These metrics identify the amount of time and energy used by the node to perform its functions in the simulations. The duration times shown in Table 1 were adopted from [20]; however, we rounded these times to the nearest millisecond because the simulation program we developed is based on one millisecond increments.

The Receive-Transmit (RX-TX) requires the node to switch from receiving to transmitting in order to transmit the frame and then switch back to receiving; thus, the execution of transmitting a frame requires two RX-TX phases, one to switch from receiving to transmitting and two to switch from transmitting to receiving. The RX-TX phase within [20] has a period of 0.4 ms, which totals 0.8 ms during the transmission and receiving process of a frame. Since our simulation program's smallest increment in time is 1.0 ms, 0.8 ms is rounded to 1.0 ms and is used to represent the two RX-TX switching phases detailed in Table 1.

To complete a frame transmission, the node must execute the following phases, totaling a period of 7.0 ms: CSMA, RX-TX switch, transmit, and RX-TX switch.

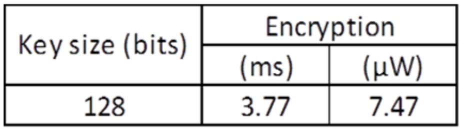
Table 1. Phases of Node Operation. Adapted from [20].

Phase	Power Draw	Duration
Wakeup	20 mW	1 ms
Pre-process	24 mW	4 ms
CSMA-CA	72 mW	2 ms
Transmit	90 mW	4 ms
Receive	72 mW	4 ms
RX-TX switch	54 mW	1 ms
Post-Process	24 mW	1 ms
Waiting	72 mW	Varies
Go to Sleep	20 mW	1 ms

Multiple logs record the status of the node throughout the lifespan of the network. One log only records each time a node transitions from one phase to another, while another log tracks the status in which the node is currently, and a third log is used to record the node's time in phase by the millisecond. The node's status logs are critical since they are used by multiple modules within the simulation program. The node's status logs include the expiration time for the phase in which the node is to prevent the node from performing/entering another phase. In most simulation programs, detailed status change logs and current node status logs are not available to the researcher, limiting a researcher's ability to troubleshoot any errors. As this network is meant to be deployed by the USMC, the ability to troubleshoot is imperative; thus, we provide the researcher with the pertinent logs. These logs are further discussed in Section G of this chapter.

Within [20], encryption was not addressed. We assumed encryption was not incorporated into the power and time metrics given in Table 1; therefore, we incorporated the time and power draw for encryption into the simulation. In [21], metrics for various brands of sensor nodes performing multiple methods of encryption with varying key sizes are given. To give the best results, we chose to use the TelosB platform since it is more power efficient [21]. Within the TelosB platform, we used the power draw and times obtained from the implementation of AES 128 to match our WSN model. The power draw and time metrics with AES 128 incorporated are shown in Table 2.

Table 2. Power Draw and Duration Time to Perform AES-128 Encryption. Adapted from [21].



Key size (bits)	Encryption	
	(ms)	(μW)
128	3.77	7.47

The nodes within the simulation are only awake for the time period in which there is activity. If a node does not have any activity for 25.0 ms, it transitions to the sleep phase. The duration of 25.0 ms was chosen to compensate for transmissions occurring when the node is processing a received frame from a node that is hidden from the transmitting node. The duration of 25.0 ms accounts for the time between the third and fourth transmissions, CSMA-CA, and both the RX-TX switch phases between actual transmissions of a frame, allowing the follow-on node to see the retransmission before entering its sleep phase.

There may be instances when the transmitting node cycles through both the primary and secondary routes without a confirmation of the next hop node transmitting the frame. In this situation, the transmitting node stores the packet and waits 1.0 s plus a seed number between 1 and 200. This provides enough time for the network to clear any data congestion, transmit all the frames to the BS, and allow all the nodes to enter their sleep phase.

C. FRAME PARAMETERS

As with a node, the frame also transitions through multiple phases. The phases include creation (unprocessed), transmission, processing, and completion. A log is also maintained, which tracks the status of the frame as it transitions between phases. Within the log, a phase expiration is attached to each node's entry and coincides with the duration metrics used for the nodes.

D. NETWORK PARAMETERS

As previously discussed, the network implements a modified CSMA-CA BEB protocol with an additional seed number added to the BEB for frames transiting from the neighboring nodes to the BS. The primary focus of this section is to discuss the modified CSMA-CA BEB and the parameters used. We also discuss the addition of the seed number for the nodes forwarding traffic to the BS via an extra relay node.

1. Modified CSMA-CA BEB

CSMA-CA BEB is a protocol that determines a set time period between transmissions of the same frame to avoid collisions with other transmissions from other nodes. Once the initial time period is determined, each retransmission doubles the previous time period between the transmitted frames [17]; however, this model is used within the IEEE 802.11 standard and assumes that each node is awake and receiving the frame [17]. Within the simulated network, when the nodes are no longer forwarding or sensing other events, they go to sleep to conserve energy. This sleep feature requires the transmitting node to send a frame to wake-up the next hop node. Even though it only takes 1.0 ms for a node to wake up, the transmitting node must take into account that the next-hop node might already be awake and receive the frame successfully; thus, the time the transmitting node must wait before beginning the second transmission process is 6.0 ms, which permits receipt of the next-hop node's transmission.

After the second transmission, the typical BEB takes place with the initial back-off period of 10.0 ms, then 20.0 ms after the third transmission and 40.0 ms after the fourth transmission. If forwarding of the frame has not been detected after completing the fourth transmission, the transmitting node switches the frames next-hop destination to the secondary route. The transmitting node then performs the same sequence again while simultaneously adjusting the designated path bits within the header.

2. Seed Number

The CSMA-CA BEB protocol also provides a method to alleviate network congestion in the node by retransmitting at different set time intervals [17]. To prevent the occurrence of two nodes transmitting at the same time near the BS, a seed number between 1.0 and 4.0 ms is added to the back-off time. The seed number creates disparity between the back-off times between transmitting nodes. With the application of the seed number, neighboring nodes that initially transmit at the same time has a smaller probability of retransmitting at the same time since each node applies a random seed number to the BEB. This helps mitigate congestion caused by collisions as the network begins to taper to the BS.

E. SIMULATION PROGRAM

To replicate the network, the simulation must be able to mimic multiple frames transiting the network at one time. Multiple parts of the network must also be able to function simultaneously. Given these factors into consideration, the driving mechanism of the program is time based. For every iteration, each function is cycled through the network while logs track each node’s status, time in phase, frame status, and the creation of new events within the network. In the following subsections, the different modules that were programmed in MATLAB to deal with the various network functions performed within the simulated environment are discussed.

1. Main

The cornerstone of the simulation program is the module, Main. In Main, the network is created when multiple modules assign user supplied files to simulate an already deployed network. The files contain the routing table, node addresses, events to simulate, and empty logs for the simulation. Logs created within the simulation include a PCAP log, a transiting packets log, and each node’s time in phase log. Within the Main module, the researcher may select which event table to use during the simulation. The event table provides sensor detections which simulates a series of real-life events that trigger a node to create a frame which is subsequently transmitted to the MS. Main is also where the researcher can select the type of attack and determine which node the attack targets within the network. This information is then used by the other modules within the simulation program.

2. Run Simulation

Within Run Simulation, the timer begins at 0.0 ms and increases by 1.0 ms after cycling through all functions. The timer appends a 10.0 s cushion after the last event to ensure the network is clear and all retransmissions have been completed. Without an added cushion, the nodes within the network are not able to complete all phases; the time cushion ensures the network is free from active packets. Once the simulation time is computed, the simulation begins cycling through the major functions 1.0 ms at a time. These functions mimic multiple aspects of the network, which including detecting collisions, ensuring the nodes and frames transition to the next phase, the creation of new packets due to a triggered event, and the tracking of the nodes phase status to calculate its total power draw.

3. Check For Errors

The Check For Errors module is one of the sub modules within the Run Simulation module. It cycles through all open frames within the network. The module specifically looks for nodes that are transmitting and compares them to a precompiled list of nodes that affect the receiving node. If a neighboring node of the receiving node is also transmitting, a collision occurs and data is not correctly received. The affected frames are then marked with an error and are not successful during the processing of the frame, resulting in retransmission from the transmitting node.

4. Check Node Status

The next sub module within the Run Simulation module is the Check Node Status module. This module performs the necessary transitions to ensure each node transitions to its next phase at the appropriate time as the nodes within the network cycle through different phases. The Check Node Status module checks the node’s current status and compares the phase expiration time within the log and the current time. If the phase expiration time is reached, the node’s status is changed by removing the node from its current phase and placing it in a waiting phase; this allows the node to transition into another phase in the following modules. Multiple logs are maintained to record each phase transition of the nodes; however, the transition is not yet recorded in the time in phase log since the node may transition into another phase in later modules.

5. Check For Open Packets

The Check For Open Packets is also a sub module contained within the Run Simulation module. It cycles through all of the active or open frames within the network, checking the phase expiration times attached to each frame. When the phase expiration time of the frame is over, the next phase to be executed for the frame is initiated. This module is used to perform the processing of a new or received frame and transmit the packet. Before executing the next phase, the node status is also checked to ensure the node is available to perform the next phase of the frame. The following are sub modules contained within the Check For Open Packets module.

a. Transmission

The main function of the Transmission module is to simulate the frame's transmission within the network. The module checks the neighboring nodes to determine if one is transmitting by listening to see if the channel is clear, a key piece of the CSMA-CA protocol implemented within the simulation. After determining the neighboring nodes are not transmitting, the node then transitions into a transmitting state. At this point, the transmitting node transitions and the module cycles through all of the neighboring nodes while performing the necessary node transitions.

b. Processed Received Packet

The Processed Received Packet module contains a simulation of bit errors in which one bit error is inserted for every 100,000 bits. The bit errors are executed by a function that randomly inserts the bit error, so this has the potential to occur anywhere within the frame. The frame is then checked for the correct addresses, and the frame's CRC is calculated and compared to the transmitted CRC to determine if any errors were observed. The CRC code was obtained from [22] and modified to fit the CRC length required by the network. If the frame passes all of the checks, the frame’s header is appended with the addresses required for the next transmission.

6. New Events

The New Events module (also a sub module of the Run Simulation module) checks the event table for the established time threshold at which event detection occurs. Once the time threshold is reached, a data frame is created. To prevent skipping over the triggered detection, the simulation references the frame’s status log, which ensures that the frame cycles through each phase along with the affected node. The encryption of the frame and the appending of the MIC are also performed within this module. The code for the encryption was obtained from [23] and modified to perform both the encryption as well as MIC within the simulation. Finally, the module checks the status of the triggered node to determine if and when the transition must take place. The New Events module must be the last transitioning stage due to the fact that the node continues until it has completed the current process before entering the waiting phase.

7. Check For Energy Use

At the end of every cycle within the Run Simulation module, each node’s current status is tallied and entered into the node time in phase log. This log is then utilized to determine the power draw for each node during the simulation timeframe.

8. Attack Modules

There are three different types of attacks conducted within the simulation program, and each has its own module. Subsections of code have been added within the other modules to facilitate simulation of the following attacks.

a. Spoofing

In the spoofing attack, the node selected by the researcher is imitated, and spoofed frames are injected into the network at the next-hop node. The intent of the spoofed node within this thesis is to inject frames with incorrect information while blending in with network traffic. To blend in with the network traffic, the frame injection occurs when the receiving node is still awake, which can be done by the rogue node observing the receiving node's transmission behavior. By only injecting when the node is awake, we can infer that an event detection has occurred within the WSN. Since the injected frame transits at nearly the same time as the traffic generated from valid transmissions, the injected frame does not stand out. Considering the frames are injected when the receiving node is awake, the number of injected frames is dependent on the receiving node's activity and varies between each simulation trial.

To simulate the rogue node's monitoring of the receiving node, the rogue node checks the receiving node's status every 100.0 ms. If the receiving node's status is in the waiting phase and none of the neighboring nodes are transmitting, the rogue node injects a frame. After injecting the frame, the rogue node does not perform another injection until 1.0 s later.

b. DOS

In the DOS attack simulation, the researcher selects a node to be affected. To simulate the attack, a rogue node that is constantly transmitting frames is placed near the desired node and noted by the researcher within the transmission file. The rogue node's continuous transmission forces the receiving node's status to stay in the receiving phase. While in the receiving phase, the affected node does not perform any other functions within the network, rendering the node incapacitated.

c. MITM

The MITM attack requires the researcher to select two neighboring nodes since the attack occurs during the frame transmission. Once the nodes are selected by the researcher, a frame sent by an affected node is changed prior to the processing of the receiving node. The change of the frame between the transmission and processing phases simulates a rogue node between the two nodes performing a MITM attack. Unlike the Spoofing module, the MITM attack is performed on each transmitted between the two affected nodes.

9. Display

The Display module displays the results and a log analysis after the conclusion of the simulation. The analysis is similar to what the MS performs in an actual deployed network. The analysis includes determining whether secondary routes were used for each node and the number of frames sent by each node that was not authenticated. Part of the analysis performed within the simulation but not able to be performed by the MS includes the percentage of successful frames that transited the network, the amount of time each node spent within a phase, and the power draw of each node in the simulation.

F. SIMULATION USER FILES

The module Main requires user provided files to build the network prior to executing the simulation. These files setup the devices by defining their capabilities as well as the network's capabilities. Within the files, the nodes and frames also receive a reference number to simplify the process. The reference number prevents the need for a researcher to reference each node by its address and each frame by its IP address and sequence number. Within this simulation program, the reference number for each node ranges from 1 to 27 and each frame obtains its reference number in sequential order from one in whole increments until there are no more frames carrying event information.

1. Nodes (Create Nodes)

Within this simulation 27 nodes are created with node 27 acting as the BS. Each node is assigned an IP and MAC address as well as a randomly generated key that is shared with the MS.

2. Routing Table (Create Routing Table to Master)

The routing table used contains both the primary and secondary paths for all of the nodes.

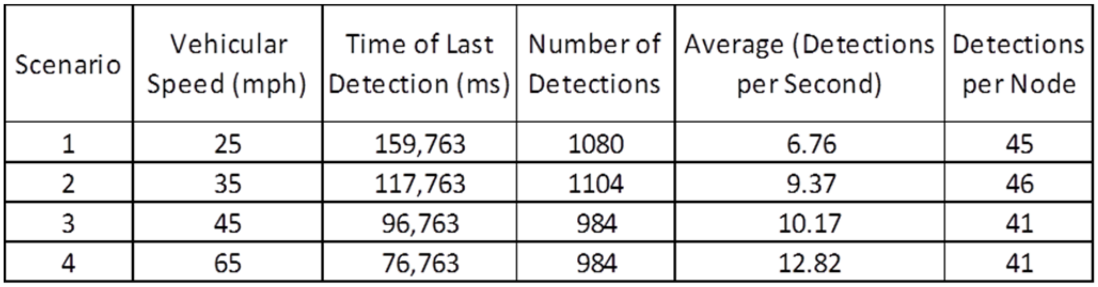
3. Affected Nodes (Create Energy Table)

To determine which neighboring nodes are affected during the transmission of a frame, the researcher provides a file containing a list of all of the nodes and which nodes it affects when transmitting.

4. Events (Create Event Table)

The event table is used to trigger the detections within the program. Each event entry denotes a time the event is triggered and the node that detected the event. The actual event is not necessary and is not recorded since it is not the focus of this thesis. Within this thesis four event tables were used. Each table focuses on a speed a vehicle would likely travel. The speeds for each of the four tables are 25, 35, 45, and 65 miles per hour (mph). Each table also contains event triggers of an individual walking alongside a street to increase the number of detections, which in turn creates more traffic within the tactical WSN. When vehicles are transiting the intersection, a spacing of 2.0 to 3.0 s is used to simulate a typical roadway environment. The parameters within each scenario are shown in Table 3.

Table 3. Parameters Used for each of the Four Scenarios that Are Simulated



Scenario	Vehicular Speed (mph)	Time of Last Detection (ms)	Number of Detections	Average (Detections per Second)	Detections per Node
1	25	159,763	1080	6.76	45
2	35	117,763	1104	9.37	46
3	45	96,763	984	10.17	41
4	65	76,763	984	12.82	41

G. SIMULATION LOGS

Mentioned throughout this chapter is the use of logs within the simulation. This is beneficial to the researcher as it provides a way to examine how the frame transits the network and aides in network troubleshooting. Similar logs are used within other well-known simulation programs such as NS3 but do not provide the in-depth network logging that this simulation program provides.

1. PCAP

The PCAP log records every transmission that is made by a node within the network. The entry contains the frame number, time at which the transmission occurred, the source node number, the destination node number, and the entire binary form of the frame.

2. Open Packets

The open packets log contains every individual frame created during the simulation. Each line within the log is assigned to a frame, and the recorded data represents the functions through which the frame transits. Each line in the log contains the frame number, phase expiration time, the status of the frame, the number of transmission attempts made by the current node, the source node number, destination node number, and any incurred errors.

3. Node Status

The node status log records the current status of each node. Each node is assigned a line within the log. Contained within the log are the node number, phase expiration time, the phase the node is in, and the sequence number of the last frame that originated from the node.

4. Node Status Table

The node status table records every transition made by a node. In this log, the node is not assigned a line, but a new entry is made when a node transitions between phases. Within each entry is the node number, the time the transition occurred, the phase the node was in, and the phase to which the node transitioned.

5. Time in Phase

Within this log, each node is assigned a line, and the time spent within each phase is recorded. This log is updated within the Check For Energy Use module, where the phase of the node is checked and the value within the respective phase for the node is incremented by one. Ultimately, the log provides the researcher with the amount of time the node spent within each phase, which is then used to determine the power draw of the node.

H. CHAPTER SUMMARY

In this chapter, the sensors used within the simulation along with their deployment methodology were examined. In addition, the node parameters used to determine power draw upon the completion of the simulation were examined in-depth. The incorporation of the phases into the frame and network parameters was then discussed. The modules within the simulation program were determined by detailing the critical modules within the program, including the test modules. The different files programmed in MATLAB to build the network were explained in depth. A detailed discussion of the series of logs that are maintained throughout the simulation was also provided.

THIS PAGE INTENTIONALLY LEFT BLANK

V. SIMULATION RESULTS AND ANALYSIS

An assessment of the proposed 6LoWPAN WSN is needed in order to validate the effectiveness of the implemented cyber security mechanisms. While the focus is on the cyber security mechanisms within a 6LoWPAN enabled WSN, it must be anticipated that these WSNs can be deployed to different types of environments. The different types of environments create varying levels of network traffic. We simulate these different environments using four scenarios that mimic the varying levels of traffic density. The scenarios developed within this thesis mimic vehicles transiting an intersection at different speeds. These scenarios then allow the researcher to determine the effects of the implementation of the proposed cyber security mechanisms within a variety of simulated network environments with each simulation listed as a trial.

The network topology used in the simulation is shown in Figure 12. A reference number is given to each node in the figure. The reference number provides a simple method for the researcher to reference a node within the program and within the user created files. There are four scenarios simulated, each representing a specific speed of vehicles traversing an intersection. The first scenario represents vehicles traveling at 25 miles per hour (mph), the second at 35 mph, the third at 45 mph, and the last scenario represents vehicles traveling at 65 mph. Each of the four scenarios are initially simulated with no attacks occurring. These trials act as the normal conditions of the tactical WSN, creating a baseline to compare the results from simulations that incorporate attacks performed on the WSN.

The nodes selected for each attack remain the same for all scenarios to allow for comparisons between the different network environments. The total number of trials conducted for each scenario is as follows: five trials with no attack, five trials with a spoofing attack on node 24, five trials with a spoofing attack on node 16, a DOS attack on node 5, a DOS attack on node 25, and an MITM attack between nodes 7 and 2. In total, each scenario has six different network implementations with five trials per implementation, resulting in a total of 30 trials for each scenario. This results in a total of

120 trials for the program. Due to space, we show and discuss only a subset of the results obtained.

In the following subsections, each attack is discussed and analyzed. An analysis of each of the attacks is displayed from the view of the MS with the implemented cyber security mechanisms in place; these mechanisms lead to either the detection or mitigation of the attack. The power draw for each node, which is not analyzed by the MS, is also discussed. Within the analysis slight variations within the results may be observed due to the implementation of frame errors. The frame errors are caused by frames being dropped or not properly received and can occur undetected on the last hop to the BS.

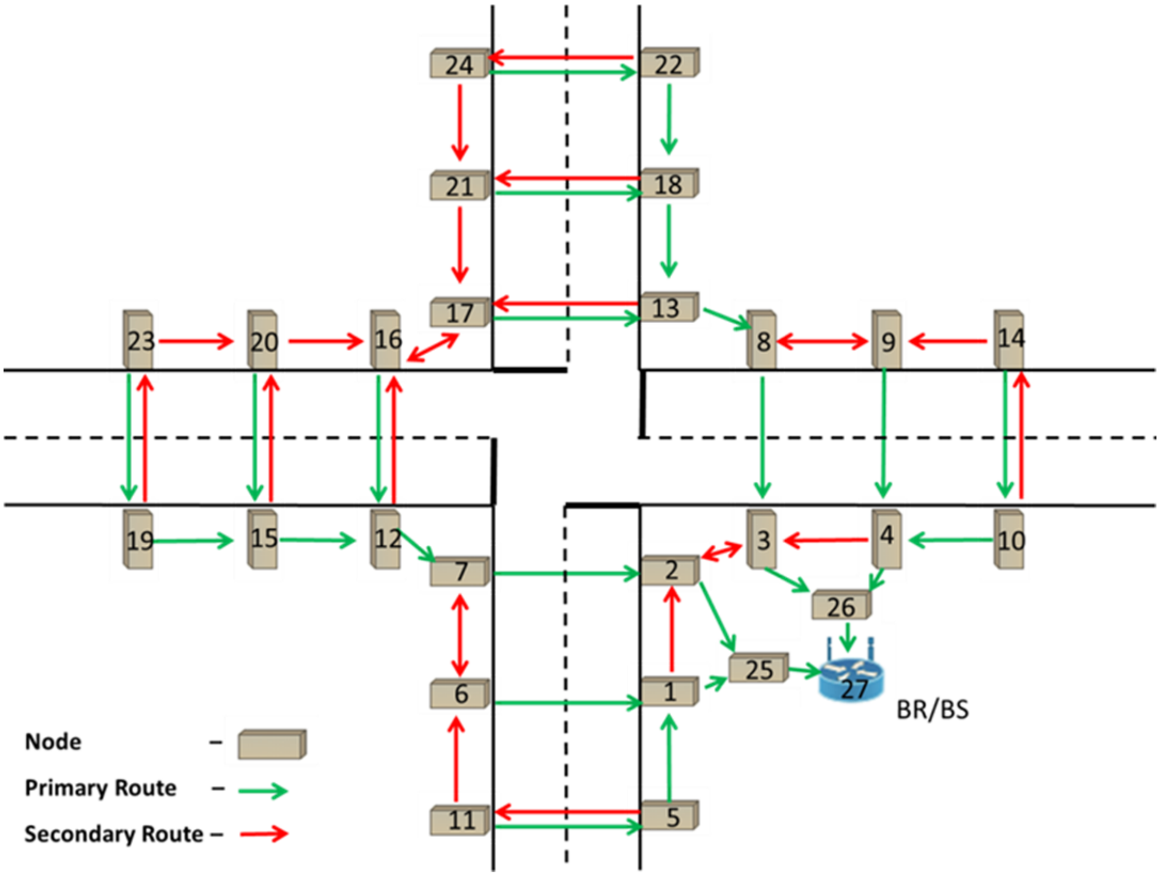


Figure 12. Assignment of Reference Numbers to the Node of the Tactical WSN

A. RESULTS

1. Spoofing

The purpose of the spoofing attack is to test the efficacy of the MIC security mechanism added to the IEEE 802.15.4 6LoWPAN enabled frame. The two nodes imitated in the spoofing attack are nodes 16 and 24. Node 16 was chosen since the surrounding nodes have a higher traffic density than a node on the edge of the network. Node 24 was selected since it is on the edge of the network with limited network traffic flow and is one of the least protected nodes in the WSN. In order to detect a spoofed node within the WSN, the MIC security mechanism is used to authenticate the frames received by the MS.

The number of spoofed frames detected by the MS and sent by the rogue node imitating node 16 is shown for Scenarios 2 and 3 in Figures 13 and 14, respectively. As expected, the number of detected, spoofed frames varies between trials due to the activity of the receiving node. The number of detected, spoofed frames also varies between scenarios. This was determined through further analysis of the logs maintained during each trial. It is observed that fewer frames were injected as the speed of the vehicles being simulated increased.

We calculate and analyze the rate of the frames injected. The average number of frames injected for the five trials performed for Scenario 2 was 43 (Figure 13), while 38 frames were injected in Scenario 3 (Figure 14). The rate for frame injection per second was computed using

$$R_{FI} = FI_{Avg} / (T_{Last\_Frame} - T_{First\_Frame}) \tag{1}$$

where R_{FI} is the rate at which a frame is injected, FI_{Avg} denotes the average number of frame injections within the scenario, and T is the time of the first and last frames within the scenario. The R_{FI} for Scenario 2 was 0.365 frames per second, resulting in an average of an injected frame every 2.74 s. The R_{FI} for Scenario 3 was 0.495 frames per second, resulting in an average of an injected frame every 2.02 s. Even though the number of injections is less as the speed increased from one scenario to the next, the rate of injections increased, which demonstrates that frames are injected at the faster speeds if the scenarios are executed for the same amount of time. The increased rate of frame injections was expected due to the increased activity level for each node.

Since the simulation program performing the analysis of the MS can only tell if the frames being processed are spoofed, the logs were examined to determine if a spoofed frame was not detected and processed. After reviewing the logs and determining which frames were spoofed, 100% of the spoofed frames received by the BS within the program were detected and denoted as non-authenticated by the MS. Within the logs, it was also noted that not all spoofed frames were received, as a small percentage were either lost or received in error by the BS.

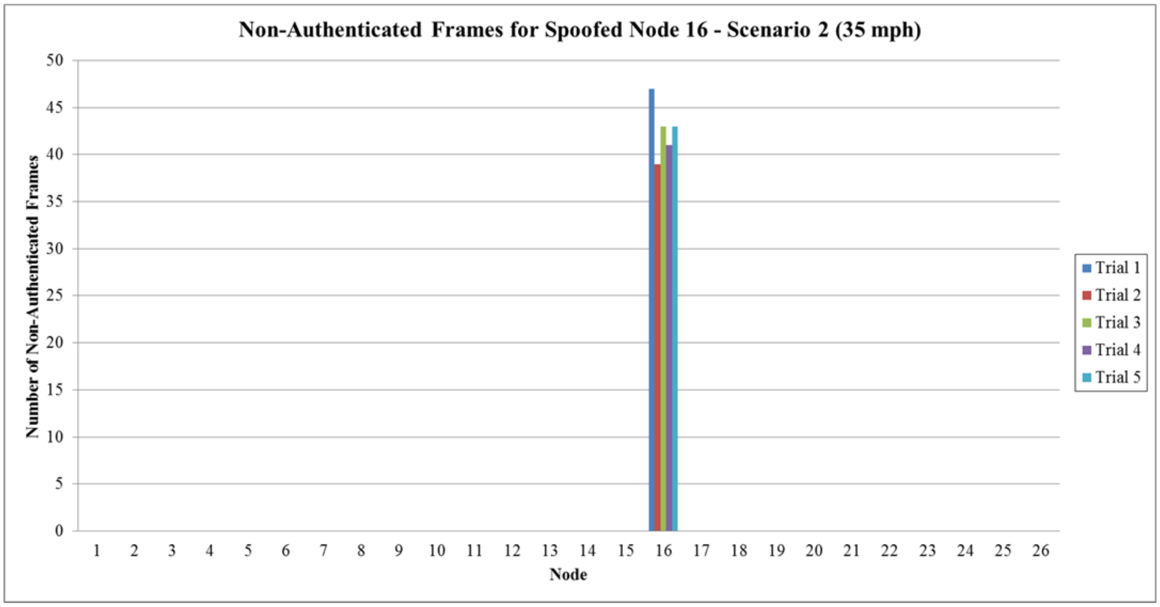


Figure 13. Number of Non-Authenticated Frames Received by the MS in each of the Five Trials for Scenario 2 Simulating a Spoofing Attack on Node 16

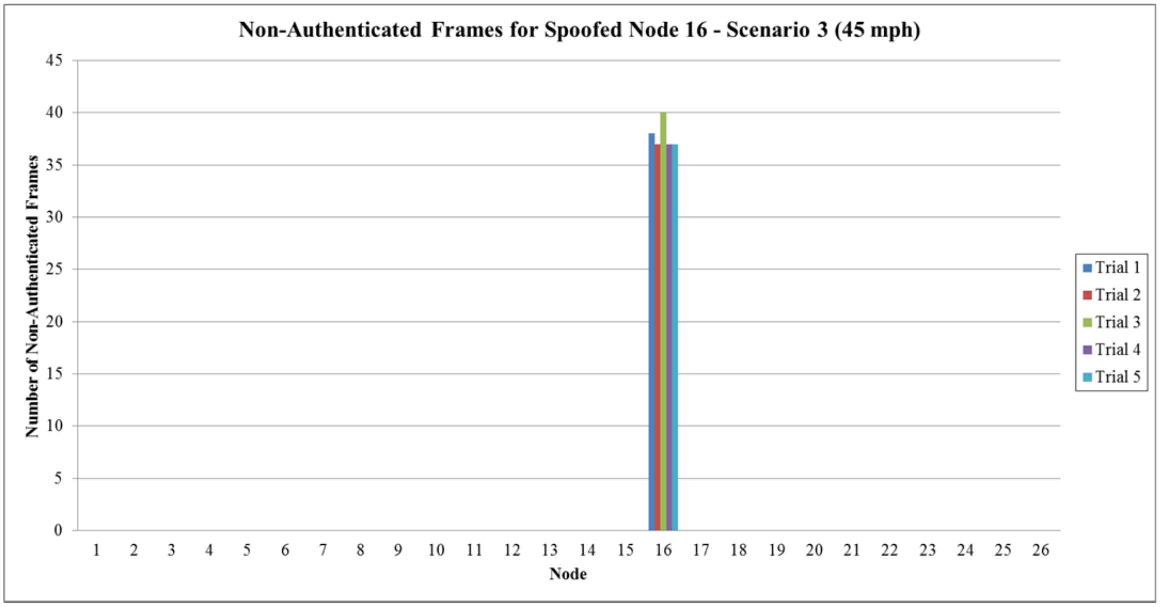


Figure 14. Number of Non-authenticated Frames Received by the MS in each of the Five Trials for Scenario 3 Simulating a Spoofing Attack on Node 16

The number of frames detected by the MS that were sent by the rogue node imitating node 24 are shown for Scenarios 2 and 3 in Figures 15 and 16, respectively. Nodes 16 and 24 share similar characteristics in the slight variation of injected frames between trials for a given scenario as well as between the scenarios; however, since node 24 is on the edge of the network and has a low density of network traffic, the node is asleep more than node 16. This explains the difference in the average number of frame injections. The average number of frames injected for the five trials performed on Scenario 2 was 26 (Figure 15) while Scenario 3's average was 23 (Figure 16). The calculated rates were also similar to those for node 16; for node 24 for Scenario 2, R_{FI} was 0.221 frames per second, resulting in an injected frame every 4.52 s. The R_{FI} for Scenario 3 was 0.238 frames per second, resulting in an injected frame every 4.20 s.

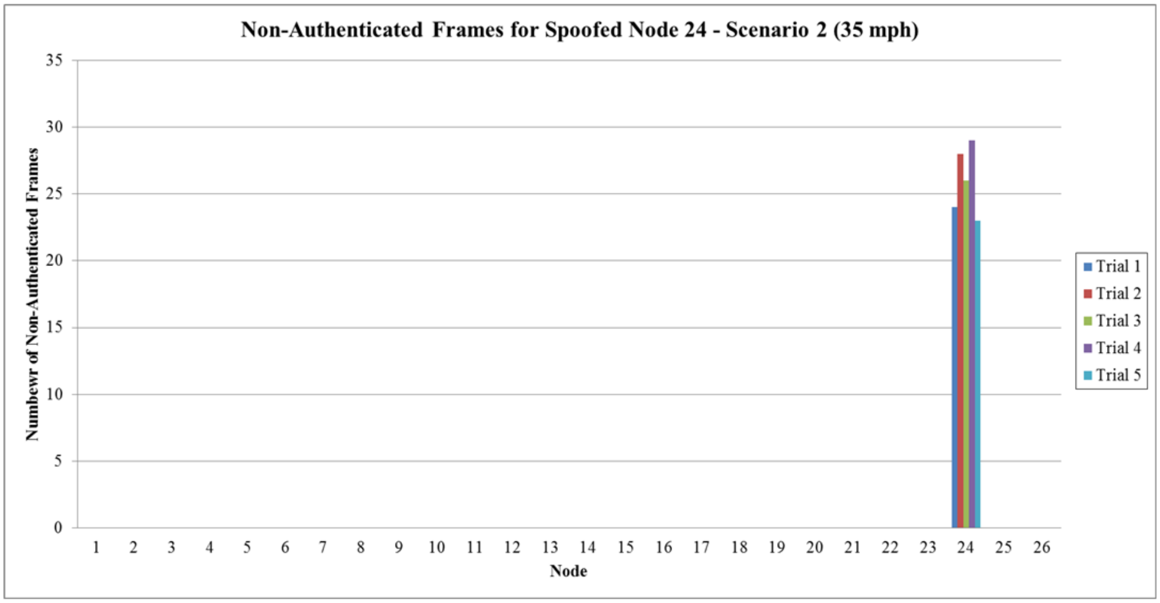


Figure 15. Number of Non-authenticated Frames Received By the MS in each of the Five Trials for Scenario 2 Simulating a Spoofing Attack on Node 24

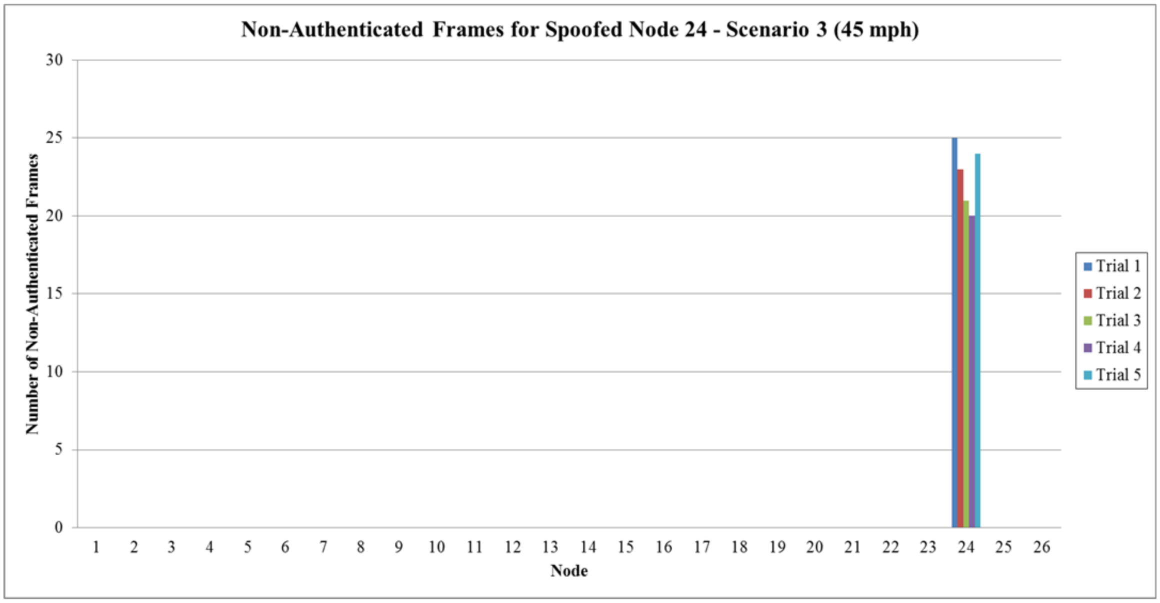
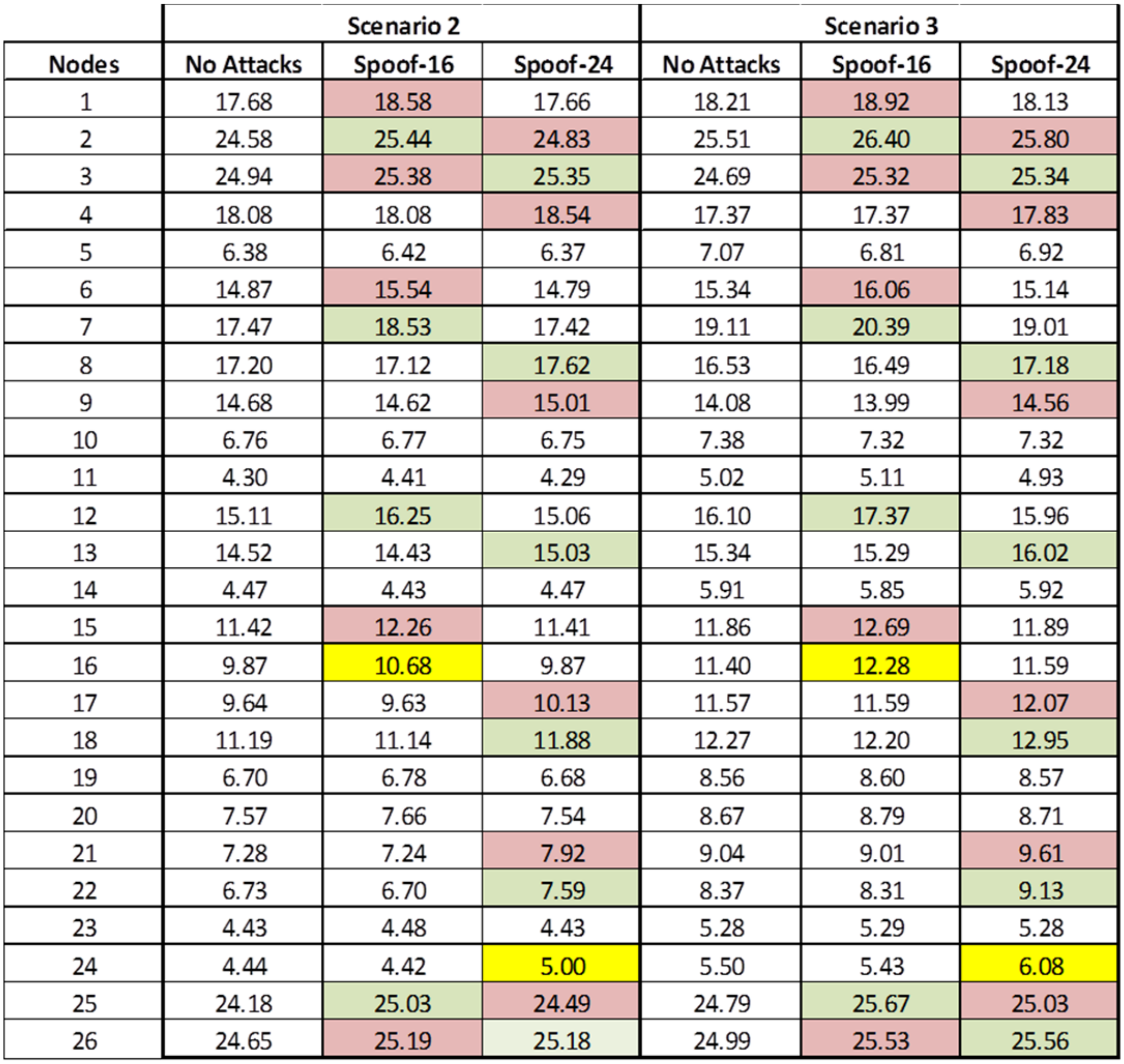


Figure 16. Number of Non-authenticated Frames received by the MS in each of the Five Trails for Scenario 3 Simulating a Spoofing Attack on Node 16

The average power draw for each network environment is displayed in Table 4. The spoofed node is highlighted yellow, the green shaded results signify the primary route used by the spoofed node to send a frame to the MS, and the red shaded results signify neighboring nodes along the primary route. As expected, the nodes that are on the primary route (green shaded) to the MS exhibited a minor increase in power draw as well as the neighboring nodes (red shaded). The increase in power draw is expected because the neighboring nodes are affected by the spoofed frames. This increases the number of frames being transmitted by the primary route nodes, causing the neighboring nodes to receive and process the increased number of transmitted frames. The small power draw increase is noticeable but would only cause a minimal impact on the affected node’s lifespan if the spoofing attack is mitigated.

Table 4. Power Draw in mW for the Spoofing Attacks on Nodes 16 and 24



	Scenario 2			Scenario 3		
Nodes	No Attacks	Spoof-16	Spoof-24	No Attacks	Spoof-16	Spoof-24
1	17.68	18.58	17.66	18.21	18.92	18.13
2	24.58	25.44	24.83	25.51	26.40	25.80
3	24.94	25.38	25.35	24.69	25.32	25.34
4	18.08	18.08	18.54	17.37	17.37	17.83
5	6.38	6.42	6.37	7.07	6.81	6.92
6	14.87	15.54	14.79	15.34	16.06	15.14
7	17.47	18.53	17.42	19.11	20.39	19.01
8	17.20	17.12	17.62	16.53	16.49	17.18
9	14.68	14.62	15.01	14.08	13.99	14.56
10	6.76	6.77	6.75	7.38	7.32	7.32
11	4.30	4.41	4.29	5.02	5.11	4.93
12	15.11	16.25	15.06	16.10	17.37	15.96
13	14.52	14.43	15.03	15.34	15.29	16.02
14	4.47	4.43	4.47	5.91	5.85	5.92
15	11.42	12.26	11.41	11.86	12.69	11.89
16	9.87	10.68	9.87	11.40	12.28	11.59
17	9.64	9.63	10.13	11.57	11.59	12.07
18	11.19	11.14	11.88	12.27	12.20	12.95
19	6.70	6.78	6.68	8.56	8.60	8.57
20	7.57	7.66	7.54	8.67	8.79	8.71
21	7.28	7.24	7.92	9.04	9.01	9.61
22	6.73	6.70	7.59	8.37	8.31	9.13
23	4.43	4.48	4.43	5.28	5.29	5.28
24	4.44	4.42	5.00	5.50	5.43	6.08
25	24.18	25.03	24.49	24.79	25.67	25.03
26	24.65	25.19	25.18	24.99	25.53	25.56

The ability of the MS to perform an analysis on received frames to determine if there is a possible spoofing attack within the WSN, as well as to determine which node to remove from the WSN to prevent further attacks, is illustrated in Figures 13 through 16. The analysis focused on the use of the implementation of the MIC security mechanism to authenticate valid frames sent by the nodes. Variations were also visible within the results between the trials and scenarios being run, but they were expected and were accounted for in the network logs. The assumption that the frame would continue to transit the network to the MS before being detected as well as the small increase in power draw along the spoofed frames' primary route to the MS were also confirmed by the results in Table 4. Overall, the MIC security mechanism is able to provide data integrity by allowing the MS to authenticate each received frame. In addition, while the increase in the number of frames transmitted has an impact on the power draw, it is minimal, keeping the WSN functional.

2. DOS

The DOS attack was used to determine if the centralized routing scheme and the use of the path indication bits could detect an attack or incapacitated node. The two nodes for the DOS attack are nodes 5 and 25 due to their differing characteristics within the WSN. Node 5 is on the edge of the network with limited network traffic flow and is also able to be physically accessed undetected by an individual due to the MAGID coverage. The implementation of the DOS attack on node 5 is displayed in Figure 17. Node 25 is one of the extra relay nodes deployed near the BS and handles half of the traffic within the network. An attack on Node 25 demonstrates the ability of the WSN to remain reliable if an extra relay node is incapacitated. Specifically, a DOS attack on node 25 validates the network's ability to utilize the secondary route of the centralized routing mechanism. In addition, it also validates the WSN's capability to accommodate an increased traffic load. Scenario 1 and Scenario 4 are used to compare the DOS results.

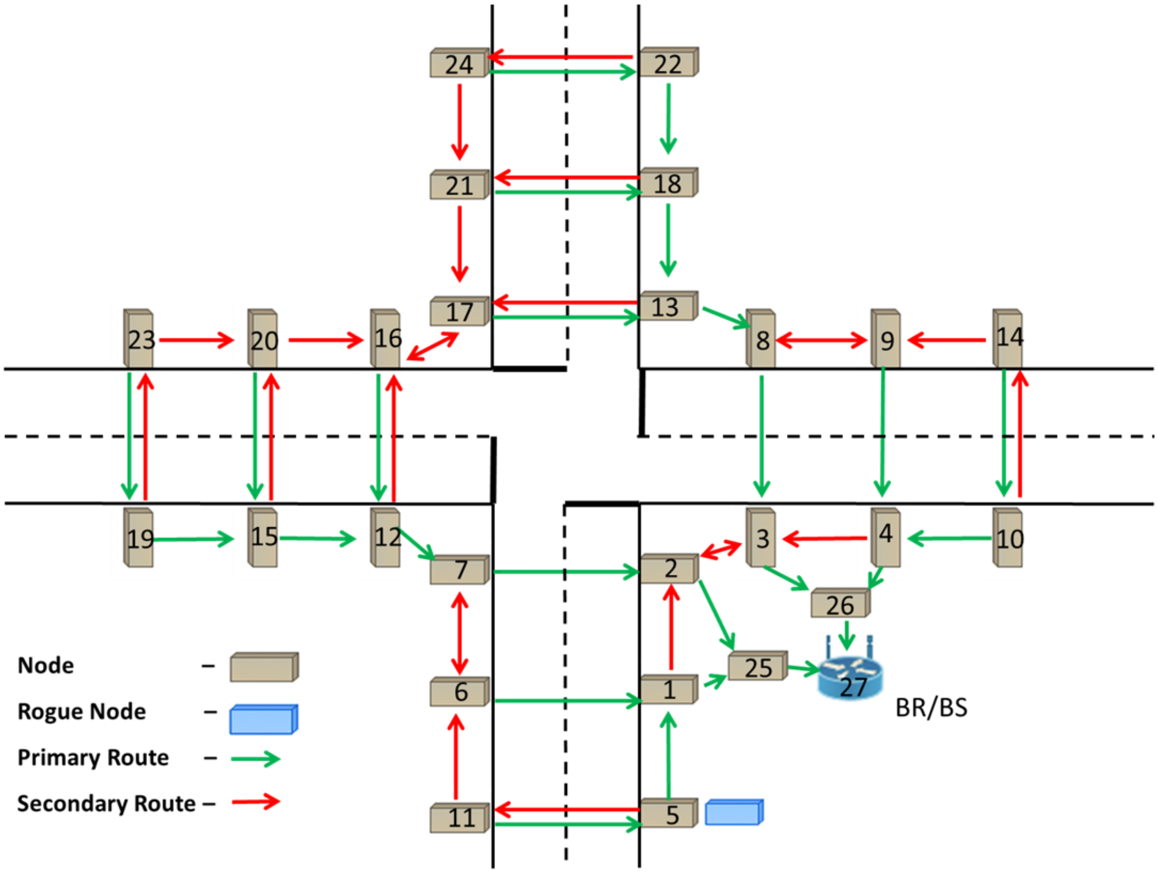


Figure 17. Deployment of a Rogue Node Performing a DOS Attack on Node 5

a. Original Hop

Our analysis of the DOS attack examines the centralized routing mechanism executed by the MS. Specifically, we first look at the number of frames the originating node failed to successfully send along the primary route. These results are shown in Figure 18 for Scenario 1 and Figure 19 for Scenario 4. Nodes 1, 2 and 11 are observed to have a significant increase in number of frames transiting to the next-hop node from the originating node. The increase for nodes 1 and 2 occurred during the trials in which there was a DOS attack on node 25, and the increase for node 11 occurred during the trials in which there was a DOS attack on node 5. This is expected since the primary route for node 11 is through node 5, and the primary route for nodes 1 and 2 is through node 25.

A closer examination of Figure 18 indicates that a limited number of frames from nodes 7, 8, 16, and 17 also took secondary next-hop routes. These anomalies can be attributed to a small amount of congestion within the network as it occurs even when no attacks are executed as denoted by the results in red in Figure 18. Also shown in Figure 19 are the same anomalies that occurred in Figure 18. As shown in Figure 19, a moderate number of frames from nodes 5, 6, 7, 8 and 11 also took secondary next-hop routes. This is attributed to the increased congestion from not only the DOS attack on node 25 but also the increased density of the traffic due to the faster detection rate. The network was not able to clear the congestion quickly enough, and the congestion began to spread to nodes two to three hops from the attacked node. Even though there was a greater amount of congestion in the network, it remained secure and continued to reliably deliver frames from affected nodes with near real-time precision.

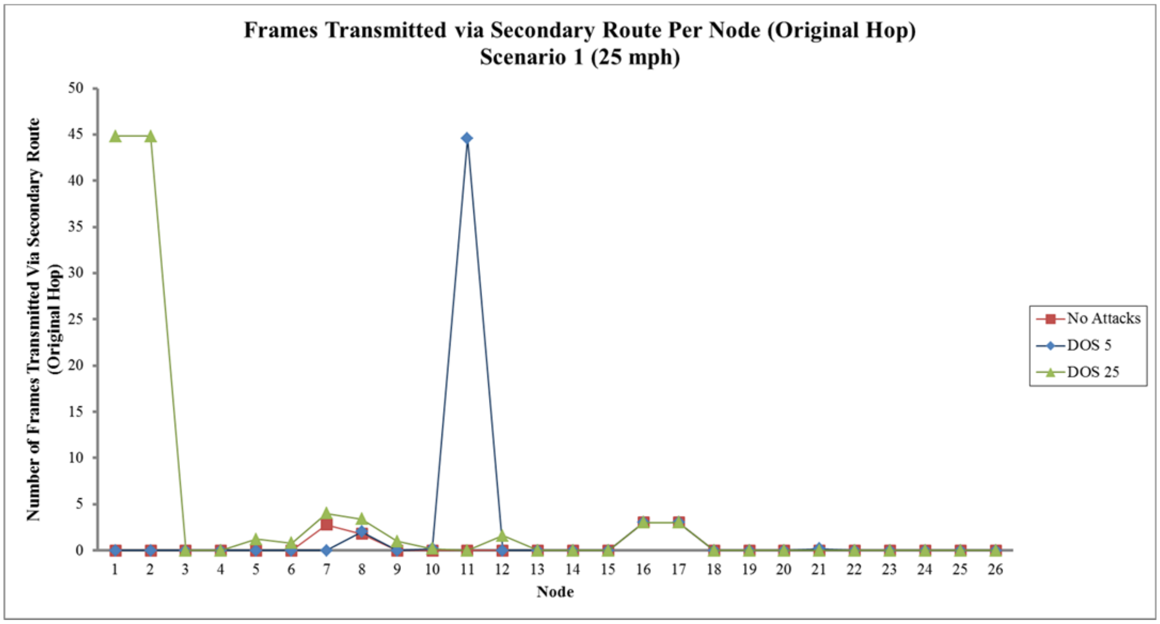


Figure 18. Frames Transmitted via Secondary Route per Node (Original Hop) in Scenario 1 (Vehicular Traffic at 25 mph) for No Attacks and DOS Attacks at Nodes 5 and 25

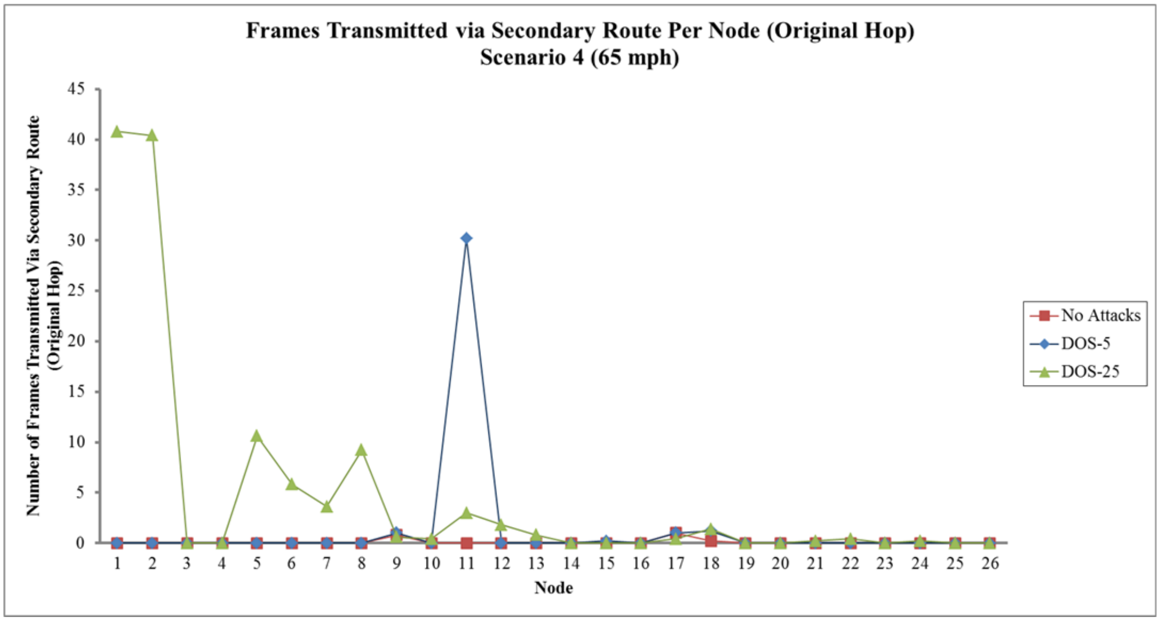


Figure 19. Frames Transmitted via Secondary Route per Node (Original Hop) in Scenario 4 (Vehicular Traffic at 65 mph) for No Attacks and DOS Attacks at Nodes 5 and 25

The analysis presented in Figures 18 and 19 is based on the implementation of the centralized routing mechanism with the utilization of the path indication bits. The path indication bits alert the MS that the frame was not able to successfully transit the WSN along the primary route from the originating node. The use of the secondary route is an indication of possible congestion or a node malfunctioning, causing the WSN to operate in a non-optimal manner. Furthermore, this increases the power draw of multiple nodes and possibly causes more network congestion. Using the information gained from the path indication bits, we see that the MS adjusts the centralized routing mechanism for optimization.

b. Follow-on Hop

The DOS attack is further examined through the analysis of the number of frames that were unsuccessfully transmitted along the primary route by the follow-on nodes. These results are shown in Figure 20 for Scenario 1 and Figure 21 for Scenario 4. As shown in Figure 20, a DOS attack on node 5 does not produce any significant changes in terms of the number of unsuccessful transmitted frames. This is expected since node 5 is only a primary route for node 11 and is not a primary route for any other nodes within the network; however, the analysis of the DOS attack on node 25 shows a significant increase in nodes that sent frames that utilized a secondary route at a follow-on node. The significant increase in the utilization of secondary routes at follow-on nodes is expected since node 25 is the primary route for half of the WSN to the BS. Minor escalations at nodes 13, 17, 18, 21, 22, and 24 can be explained by congestion experienced from the surge of network traffic utilizing the secondary routes. This is more evident in Figure 21 due to the increased density of network traffic in Scenario 4.

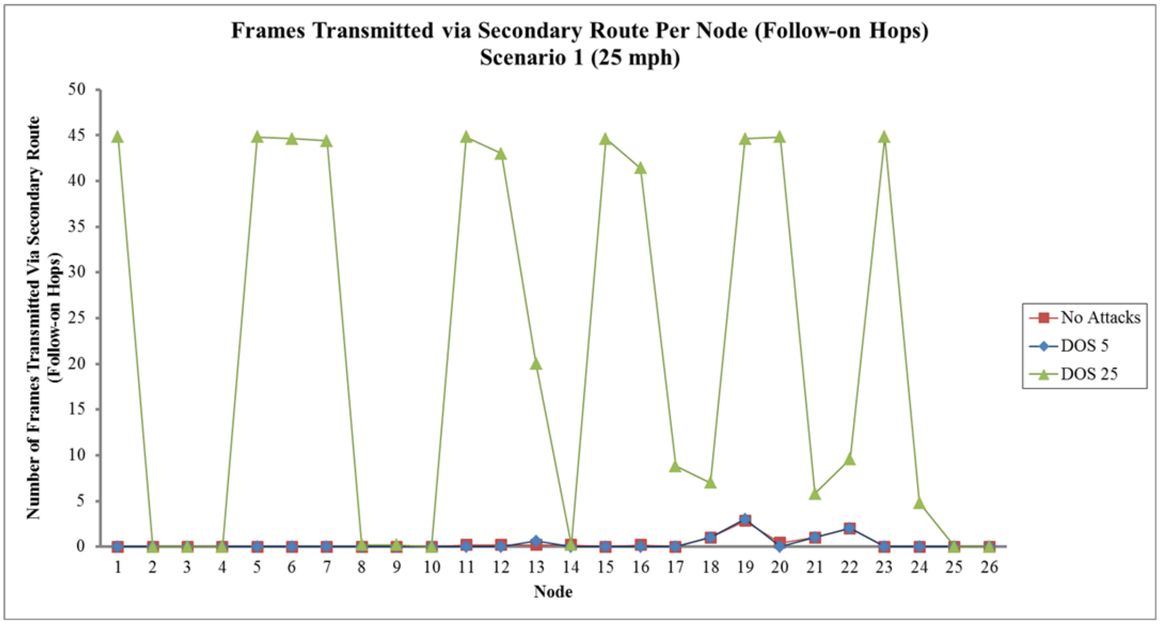


Figure 20. Frames Transmitted via Secondary Route per Node (Follow-on Hops) in Scenario 1 (Vehicular Traffic at 25 mph) for No Attacks and DOS Attacks at Nodes 5 and 25

The analysis presented in Figures 20 and 21, like Figures 18 and 19 but for follow-on hops, are possible due to the implementation of the centralized routing mechanism with the utilization of the path indication bits. The secondary route information data provides an additional awareness of the WSN's status, which the MS uses in order to determine if and/or how much congestion is within the WSN. The combination of the follow-on hop data as well as the original hop data provides a near real-time status of the WSN to the MS.

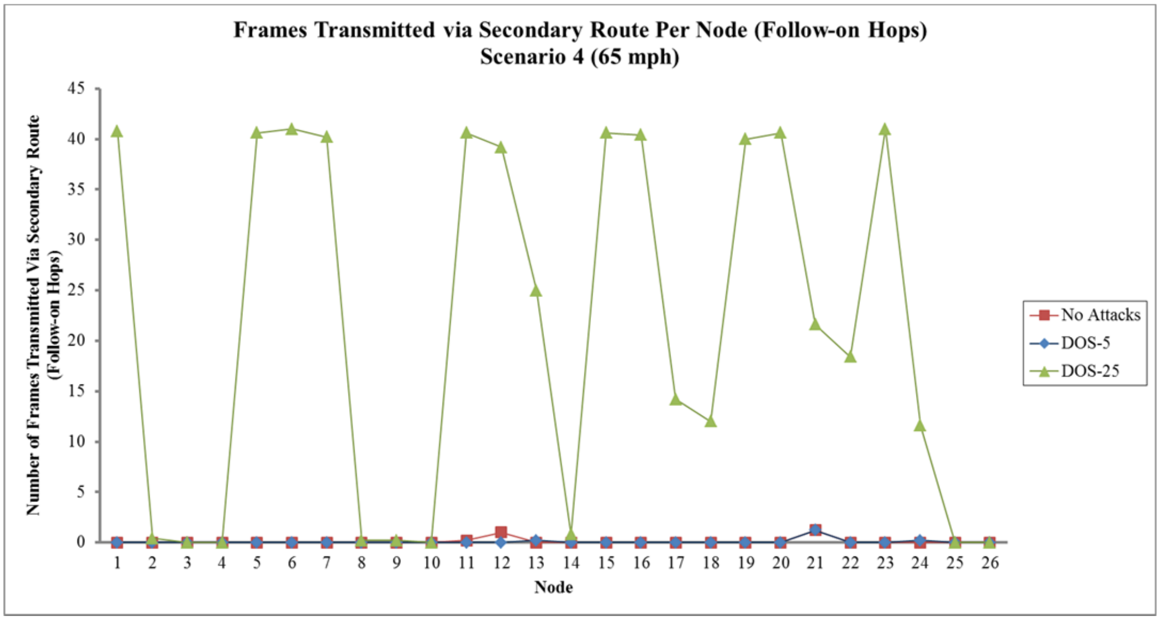


Figure 21. Frames Transmitted via Secondary Route per Node (Follow-on Hops) in Scenario 4 (Vehicular Traffic at 65 mph) for No Attacks and DOS Attacks at Nodes 5 and 25

c. *Power Draw*

The average power draw for each node during the simulations of Scenarios 1 and 4, in conjunction with either a DOS attack on node 5 or a DOS attack on node 25 as well as no attacks, are shown in Figure 22 and 23. As shown in Figure 22, there are minimal changes in the power draw between no attack and the DOS attack on node 5 trials except for node 5. The increase in power draw for node 5 is expected because node 5 is constantly receiving a signal from the rogue node performing the DOS attack. It is also observed in Figure 22 that the DOS attack on node 25 affects all of the nodes up to two hops from the BS as well as nodes 7 and 8, which are three hops from the BS. This is attributed to the increase in network traffic causing congestion, which in turn increases the power draw of the affected nodes. Similar characteristics are also seen in Figure 23, with the same nodes having an increase in their power draw for both attack simulations.

The results shown in Figures 22 and 23 have a strong correlation to the results shown in Figures 18 and 19. The nodes from Figures 18 and 19, which utilized secondary routes to transmit the frame from the original hop, correspond to the same nodes with an increase in power draw in Figures 22 and 23. The increase in the power draw for the surrounding nodes is expected since it requires the originating node to first transmit to the primary route four times prior to transmitting to the secondary route. After further examination of the results in Figures 22 and 23, we find that the nodes in Figure 23 had a much larger power draw than those in Figure 22. As mentioned before, Scenario 4, shown in Figure 23, simulates an environment in which detections occur at a rate approximately two times greater than Scenario 1, shown in Figure 22. These results are expected since the nodes have less time to sleep and spend more time awake, which increases the power draw for the nodes.

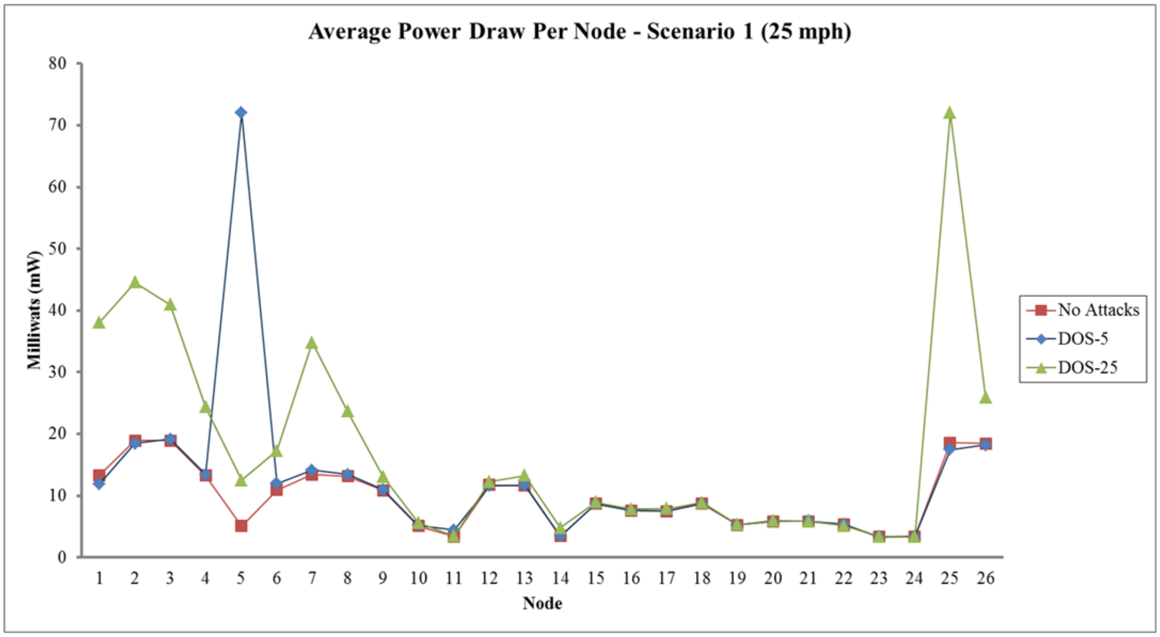


Figure 22. Average Power Draw per Node in Scenario 1 (Vehicular Traffic at 25 mph) for No Attacks and DOS Attacks at Nodes 5 and 25

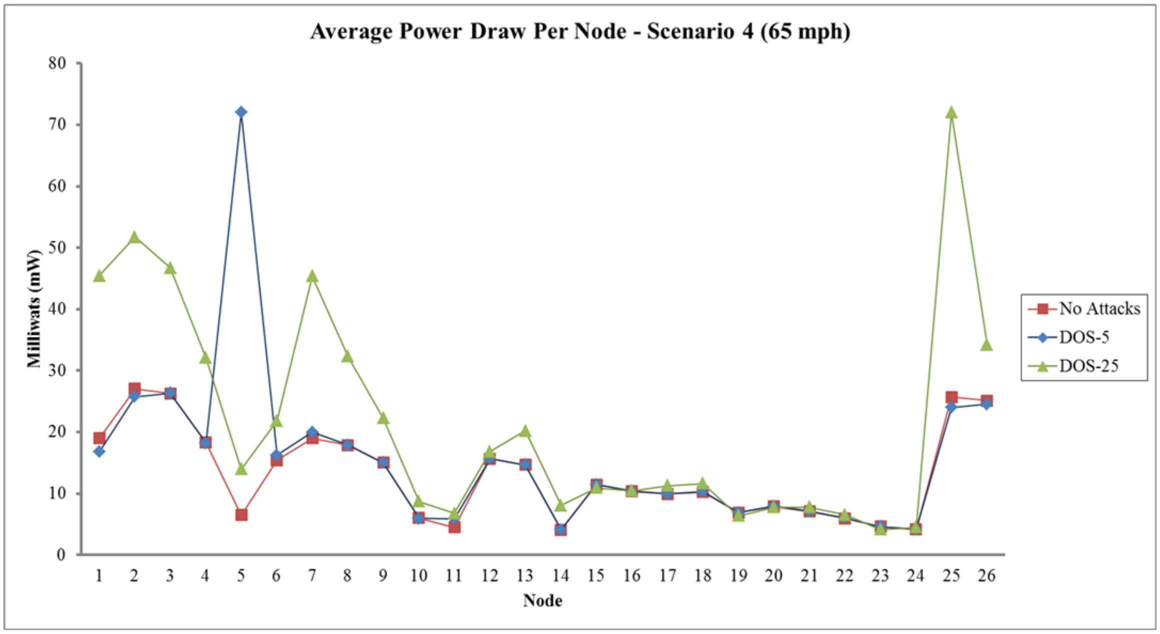


Figure 23. Average Power Draw per Node in Scenario 4 (Vehicular Traffic at 65 mph) for No Attacks and DOS Attacks at Nodes 5 and 25

3. MITM

Similar to the spoofing attack, the analysis performed on the MITM attack is focused on the implementation of the MIC security mechanism but also the centralized routing mechanism using the path indication bits. The wireless connection between nodes 7 and 2 was selected as the point of attack since it has a high density of traffic affecting a large portion of the network but not the entire network. The power draw of the nodes was not impacted as much as the Spoofing and DOS attacks. Shown in Figure 24 is the analysis of the power draw for no attacks and an MITM attack between nodes 7 and 2. The differences between the MITM simulations and no attack simulations are negligible since the power draw for each node is the same.

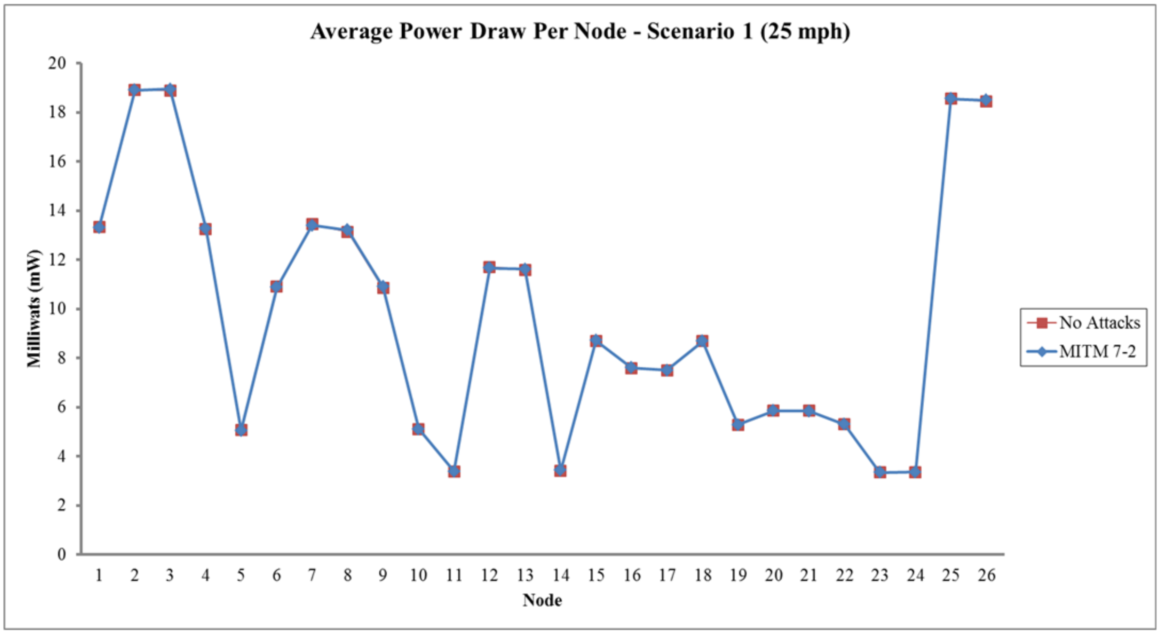


Figure 24. Average Power Draw per Node in mW when no Attack Occurs and during the MITM Attack Simulations between Nodes 7 and 2

The average number of non-authenticated frames over the five conducted trials for Scenario 1 is shown in Figure 25. Nodes 7, 12, 15, 16, 19, 20 and 23 have primary routes through node 7 to node 2, and these nodes had the most non-authenticated frames. The number of detections per node is 45 with a frame generated for each detection. A slight variation between the frames sent by nodes and the frames received by the MS is shown in Figure 25. The variation is accounted for by either a node along a frame's path utilizing a secondary route, resulting in the frame not transitioning through the affected nodes, or the frame being lost or dropped at the BS. The non-authenticated frames from nodes 17 and 18 are due to congestion within the network and the utilization of the secondary route, which ultimately went through the affected nodes 7 and 2.

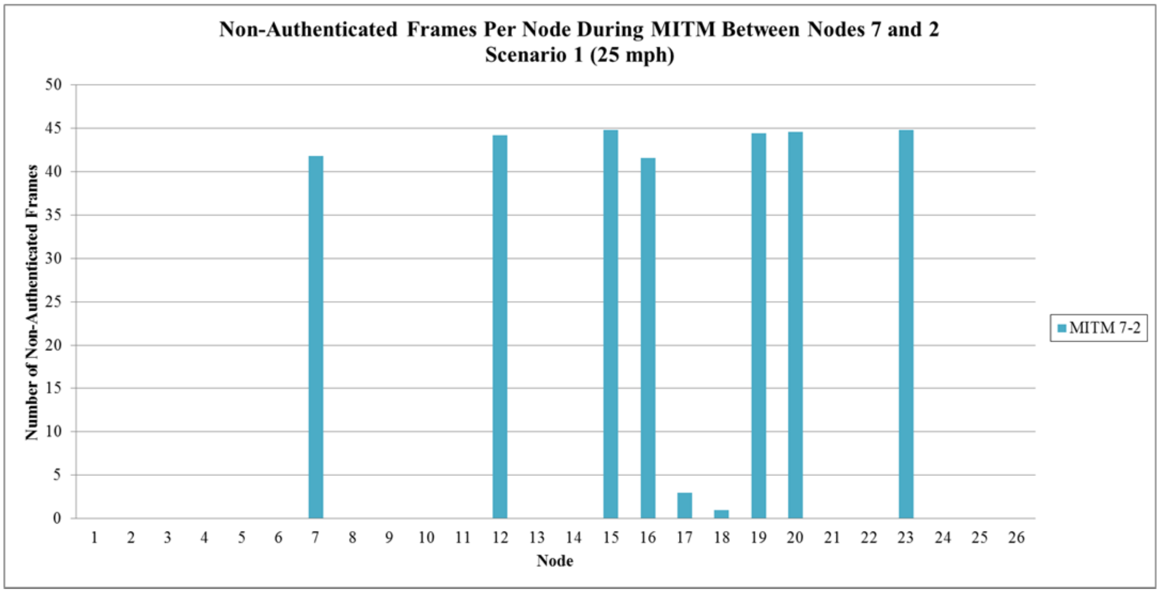
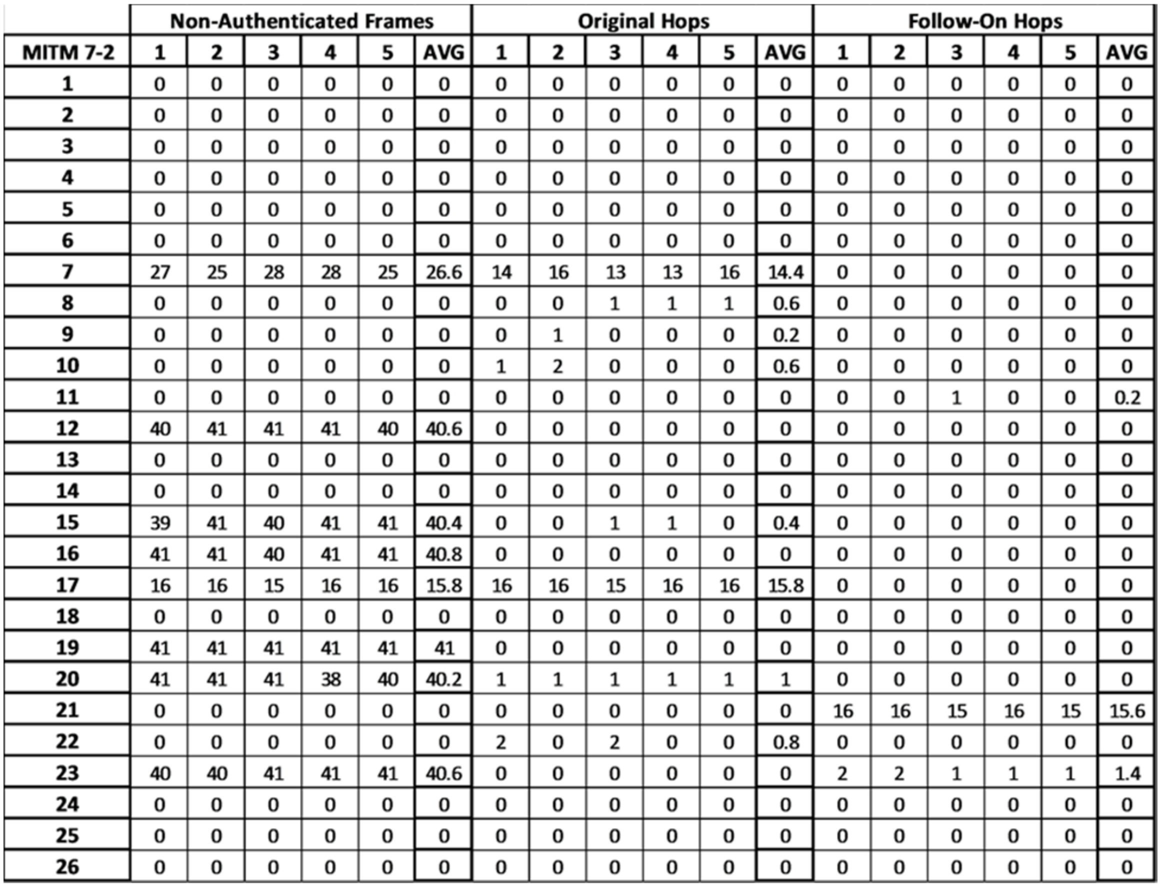


Figure 25. Number of Non-authenticated Frames Received by the MS from the Originating Node During a MITM Attack between Nodes 7 and 2

The results displayed in Table 5 are the averages of the five trials modeling the MITM attack in Scenario 3. The table has three main parameters that were quantified over each of the five trials for all 26 nodes. These parameters are 1) the number of frames that were not authenticated by the MS, 2) the number of frames that took the secondary route at the originating node, and 3) the number of frames that took a secondary route along the follow-on hops. Within Scenario 3, each node has 41 instances in which the node has a detection and transmits a frame alerting the MS to the detection.

Node 7 is significant since the number of non-authenticated frames is drastically different from the 41 frames expected to be received by the MS and not authenticated. By comparing the results of the non-authenticated frames of each node and the paths taken by the frames from the originating node, we can explain the anomaly. Node 7 had an average of 26.6 frames not authenticated by the MS. This is well below the anticipated 41 frames; however, since an average of 14.4 frames transited along node 7's secondary route, the frames were not affected by the rogue node performing the MITM attack. For instance, by adding the 27 frames that were not authenticated during trial 1 (shown in Table 3) and the 14 frames utilizing the secondary route from the originating node for trial 1 (shown in Table 3), the number of frames sent by node 7 equals 41; however, this is not the case for all of the nodes.

Table 5. Average Results of the Five Trials of MITM Attacks Conducted on Scenario 3



	Non-Authenticated Frames						Original Hops						Follow-On Hops					
MITM 7-2	1	2	3	4	5	AVG	1	2	3	4	5	AVG	1	2	3	4	5	AVG
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	27	25	28	28	25	26.6	14	16	13	13	16	14.4	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1	1	1	0.6	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	0	0	0.2	0	0	0	0	0	0
10	0	0	0	0	0	0	1	2	0	0	0	0.6	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0.2
12	40	41	41	41	40	40.6	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	39	41	40	41	41	40.4	0	0	1	1	0	0.4	0	0	0	0	0	0
16	41	41	40	41	41	40.8	0	0	0	0	0	0	0	0	0	0	0	0
17	16	16	15	16	16	15.8	16	16	15	16	16	15.8	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	41	41	41	41	41	41	0	0	0	0	0	0	0	0	0	0	0	0
20	41	41	41	38	40	40.2	1	1	1	1	1	1	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	16	16	15	16	15	15.6
22	0	0	0	0	0	0	2	0	2	0	0	0.8	0	0	0	0	0	0
23	40	40	41	41	41	40.6	0	0	0	0	0	0	2	2	1	1	1	1.4
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Node 17 shares the opposite characteristics as the node’s primary path to the MS does not transit through the affected nodes, but the node’s secondary path does. Node 17 has a secondary route from the originating node to node 16, which then goes through nodes 7 and 2. Comparing each trial within the non-authenticated frames to the secondary routes taken at node 17, we see that the results are equal.

Just like spoofing, the MS is able to perform an analysis on received frames to determine if there is a possible attack within the WSN. These results are shown in Figure 24. Unlike in a spoofing attack, where only the follow-on nodes from the spoofing attack are affected, the nodes prior to the MITM attack are the ones affected (as shown in Table 3). With the use of the MIC security mechanism to authenticate valid frames sent by the nodes as well as the routing information within the centralized routing mechanism, the MS is able to determine where the attack is occurring and which nodes to remove from the WSN. Overall, the MIC security mechanism is able to provide data integrity in that it allows the MS to authenticate each received frame. The centralized routing mechanism used in conjunction with the utilization of designated path bits helps determine which nodes to remove in order to provide network reliability.

B. CHAPTER SUMMARY

We began this chapter by describing the simplified method of referencing each node by a number in order to comprehend the results given within each subsection for the different types of attacks considered. The first attack examined was the spoofing attack that was meant to test the MIC security mechanism implementation. The second was the DOS attack that tested the detection of the attack by the centralized routing mechanism implementation. The final attack, the MITM attack, tested both the centralized routing mechanism as well as the MIC security mechanism implementations.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION AND FUTURE WORK

A. SUMMARY AND CONCLUSIONS

In this thesis, we studied the implementation of the 6LoWPAN protocol for tactical WSNs and examined the need for 6LoWPAN in tactical WSNs used by the USMC in operational scenarios. Through the use of 6LoWPAN, our aim was to reduce the manpower required to maintain the tactical WSN by allowing the WSN to be managed from a remote, secure location. Ultimately, 6LoWPAN provides automation to the data flow by eliminating the need of an individual to physically retrieve data from the COC. The 6LoWPAN protocol, with the addition of necessary cyber security mechanisms, can be implemented and used by the USMC to boost the abilities of its current WSNs.

In this thesis, we developed and discussed a comprehensive tactical WSN framework using 6LoWPAN that includes a hierarchical network design using defined network devices. The use of a structured/centralized network design allows for secure network reachability and accessibility. We implemented multiple cyber security mechanisms within the 6LoWPAN protocol. These security features included a centralized routing scheme, encryption, authentication and integrity. These features were applied/implemented into the 6LoWPAN frame structure. The combination of these various cyber security mechanisms and frame structure modifications create an effective and efficient tactical WSN that can be utilized by the USMC.

We evaluated our framework using MATLAB and tested it against three well-known attacks. Results obtained from the simulations focused on the effectiveness of the cyber security implementations and the power draw of each node. The results for the effectiveness of the cyber security implementations were derived from the metrics provided by the MS within the simulation. Specifically, the cyber security mechanisms that can be analyzed by the MS are the implementation of the MIC and centralized routing scheme with the use of indication bits.

The use of the MIC provided integrity to the WSN by preventing the authentication of 100% of the frames received by the MS in either the spoofing or MITM attacks. The use of the centralized routing scheme ensured the WSN remained functional and reliable even when one of the two nodes connecting the BS to the rest of the WSN was disabled during the DOS attack. The implementation of indication bits within the modified 6LoWPAN frame structure enabled the MS to determine that there was congestion within the network resulting from either traffic density or an incapacitated node.

The power draw for each node, which is not analyzed by the MS, was also examined. The effects of the spoofing attack demonstrated that while the injected frame within the network was not authenticated by the MS, the surrounding nodes along the frame’s path to the MS were affected with an increased power draw, shortening the node’s lifespan. The DOS attack had a much larger impact, especially on the node being attacked as well as on nodes up to two hops away, resulting in an increase in power draw either from the congestion or the utilization of the secondary route of the centralized routing scheme. As expected, the MITM attacks had little effect on a node’s power draw.

Within this thesis we provided an effective and efficient tactical WSN using 6LoWPAN that can remain reliable and secure while deployed within harsh environments. We conclude that the developed cyber security mechanisms and network structure provide a foundation on which future tactical WSNs used within the USMC can be based.

B. CONTRIBUTIONS OF THIS THESIS

The objective of this thesis was to implement a secure tactical WSN that can be deployed in a variety of environmental conditions supporting the USMC mission. The contributions of this thesis can be summarized as follows:

- Development of a command and control (administrative control) structure of the tactical WSN that incorporates node control, a unique defined/centralized routing model, and a selected keying mechanism for data confidentiality, authentication and integrity.

- Construction of a modified 6LoWPAN enabled IEEE 802.15.4 frame structure to incorporate the unique centralized routing model and selected keying mechanism.
- Enhancement of methods to aid in the deployment planning of the secured tactical WSN to prevent critical vulnerabilities.
- Simulation and evaluation of the proposed network framework against multiple attacks and testing for security robustness and energy conservation.

To the best of our knowledge, this is the first implementation of a secure 6LoWPAN enabled IEEE 802.15.4 WSN for the USMC tactical space that has been evaluated against well-known attacks.

C. FUTURE WORK

The combination of energy conservation for energy constrained devices and the implementations of enhanced cyber security mechanisms are critical for military WSN applications. The initial groundwork to achieve the security needs for the USMC tactical WSN were provided within this thesis. Future work is suggested as follows.

1. Application of Sink Node Anonymity

The theoretical framework developed in this thesis introduces a single point of failure among the deployable devices within the WSN. This single point of failure is the BS, which is a critical asset in the network. Previous research was completed on the privacy of the BS in [9]; however, cybersecurity methods were not addressed. Providing privacy to the BS with the use of the methods given in [9] and integrating it with the security framework developed in this thesis will provide another layer of defense for the critical asset and make it harder for the enemy to identify the BS and disrupt or disable the WSN.

2. Implementation of the BS

The BS is a critical element within the WSN and performs the transition between the 6LoWPAN enabled IEEE 802.15.4WSN and the selected public domain. The development and implementation of a BS that is able to perform the transitions from the

WSN to multiple types of networks within one device is vital. The BS needs to be able to connect to multiple types of public infrastructures including cellular, Wi-Fi, and Ethernet as well as other possible external military IP communication forms currently in use. The development of one device with all of the capabilities also allows for the universal deployment of the WSN, not just within a restricted location in which the designated infrastructure is present.

3. Implementation of Cyber Security on a Mobile WSN

The routing mechanisms used within this thesis is static due to the nature of the devices being used; however, the use of WSNs within a mobile environment is also of value to military applications. The WSNs can be attached to individuals, providing valuable information in regard to the individual’s location or biometric readings enhancing situational awareness of the deployed force. Neighbor discovery methods and authentication were not examined within this thesis but would be required for the implementation of a mobile WSN since nodes could connect within the WSN at any point. The determination of how the nodes could reliably discover each other as well as the development of a keying mechanism to provide authentication are required. The routing mechanism also needs to be adjusted continuously among the deployed devices to facilitate the organic nature of the network, as it changes with formation of the deployed forces.

APPENDIX. SIMULATION PROGRAM

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Main %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc;
clear all;
close all;
format compact;

% Create globals I want to use and retain for the final product

global Packet;
global Packets;
global Open_Packets;
global PCAP;
global Timer;
global Events;
global Routing_Table;
global Node_Address;
global Node_Status;
global Node_Status_Table;
global Energy_Use;
global Packet_Retrans_Tracker;
global Keys;
global Payload;
global MAC;
global Initial_Payload;
global MAC_Check;
global MAC_F;
global Payload_Out;
global Selection;
global NodeTX;
global Enc;

% Variables for conducting an attack
global DDOS_Node;
global MITM_Nodes;
global Spoofed_Node;

global fileID;

% Where do you want the information to be stored for each run?
fileID = fopen('results/test.txt','w');

% Which Scenario would you like to run?
Selection = 1;

% Build the framework
Create();
```

```
% Directed Denial of Service
% To activate the DDOS attack, uncomment and select which node you
% would like to perform the attack on.
% To simulate the attack the node will not be able to send or
% receive packets, therefore the node's status will be set to a level
% that it will not be able to interact with the deployed network.
% Node to be attacked (must be between 1 and 26
DDOS_Node = 28;
if DDOS_Node < 28
    DDOS(1);
end

% Spoofing attack
% Packets will be sent from a separate node pretending to be a node
% within the network. The spoofing node will use a neighbor node's MAC
% and IPaddresses as well as the CRC sequence. The packet is then sent
% to the Master Station where it would determine if it would pass the
% integrity check.
% Select a node to spoof (must be between 1-26)
Spoofed_Node = 28;
    Spoofing();

% Man in the Middle attack
% In this type of attack the contents of the packet will be modified
% while in transmission. It will make its way through until it reaches
% the Master Station, and at that point the MAC is tested and would
% fail.
 MITM_Nodes = [28,29];

% Initialize the Timer!
Timer = 0;

Run_Simulation();

Display();
fclose(fileID);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Create()
% Creates the framework for the network

Create_Nodes();
Create_Routing_Table_To_Master();
Create_Packet_Table();
Create_Energy_Table();
Create_Energy_Use();
Create_Event_Table();
```

```
Create_PCAP_Table();


end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create_Nodes %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Create_Nodes()
% Use the below information to create the node for your architecture
global Node_Address;
global Node_MAC;
global Node_Location;
global Node_Status;
global Wait_Time;
global Node_Status_Table;
global Keys;
global NodeTX;
global NodeRX;

% number of transmissions per node
NodeTX = [
    1       0;
    2       0;
    3       0;
    4       0;
    5       0;
    6       0;
    7       0;
    8       0;
    9       0;
    10      0;
    11      0;
    12      0;
    13      0;
    14      0;
    15      0;
    16      0;
    17      0;
    18      0;
    19      0;
    20      0;
    21      0;
    22      0;
    23      0;
    24      0;
    25      0;
    26      0;
    27      0;
    ];

%number of receptions per node
NodeRX = .
```

```
    1       0;
    2       0;
    3       0;
    4       0;
    5       0;
    6       0;
    7       0;
    8       0;
    9       0;
    10      0;
    11      0;
    12      0;
    13      0;
    14      0;
    15      0;
    16      0;
    17      0;
    18      0;
    19      0;
    20      0;
    21      0;
    22      0;
    23      0;
    24      0;
    25      0;
    26      0;
    27      0;
    ];

%   Node#   Byte 1        |    Byte 2        |    Byte 3        |
Byte 4        |    Byte 5        |    Byte 6        |    Byte 7
|    Byte 8        |
Node_Address = [
    001 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,0,0,
0,0,0,0, 0,0,0,0,  0,0,0,1;
    002 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,0,0,
0,0,0,0, 0,0,0,0,  0,0,1,0;
    003 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,0,0,
0,0,0,0, 0,0,0,0,  0,0,1,1;
    004 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,0,0,
0,0,0,0, 0,0,0,0,  0,1,0,0;
    005 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,0,
0,0,0,0, 0,0,0,0,  0,0,0,1;
    006 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,0,
0,0,0,0, 0,0,0,0,  0,0,1,0;
    007 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,0,
0,0,0,0, 0,0,0,0,  0,0,1,1;
```

```
    008 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,0,
0,0,0,0, 0,0,0,0,  0,1,0,0;
    009 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,0,
0,0,0,0, 0,0,0,0,  0,1,0,1;
    010 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,0,
0,0,0,0, 0,0,0,0,  0,1,1,0;
    011 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,0,
0,0,0,0, 0,0,0,0,  0,0,0,1;
    012 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,0,
0,0,0,0, 0,0,0,0,  0,0,1,0;
    013 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,0,
0,0,0,0, 0,0,0,0,  0,0,1,1;
    014 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,0,
0,0,0,0, 0,0,0,0,  0,1,0,0;
    015 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,0,1,
0,0,0,0, 0,0,0,0,  0,0,0,1;
    016 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,0,1,
0,0,0,0, 0,0,0,0,  0,0,1,0;
    017 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,0,1,
0,0,0,0, 0,0,0,0,  0,0,1,1;
    018 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,0,1,
0,0,0,0, 0,0,0,0,  0,1,0,0;
    019 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,1,
0,0,0,0, 0,0,0,0,  0,0,0,1;
    020 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,1,
0,0,0,0, 0,0,0,0,  0,0,1,0;
    021 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,1,
0,0,0,0, 0,0,0,0,  0,0,1,1;
    022 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,1,
0,0,0,0, 0,0,0,0,  0,1,0,0;
    023 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,0,1,
0,0,0,0, 0,0,0,0,  0,0,0,1;
    024 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,0,1,
0,0,0,0, 0,0,0,0,  0,0,1,0;
    025 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,0,0,
0,0,0,0, 0,0,0,0,  0,0,0,1;
```

```
    026 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 0,0,0,0,
0,0,0,0, 0,0,0,0,  0,0,1,0;
    027 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 0,0,0,0,
0,0,0,0, 0,0,0,0,  0,0,1,1;

    ];

Node_MAC = [
    001 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,0,0,
0,0,0,0, 0,0,0,0,  0,0,0,1;
    002 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,0,0,
0,0,0,0, 0,0,0,0,  0,0,1,0;
    003 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,0,0,
0,0,0,0, 0,0,0,0,  0,0,1,1;
    004 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,0,0,
0,0,0,0, 0,0,0,0,  0,1,0,0;
    005 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,0,
0,0,0,0, 0,0,0,0,  0,0,0,1;
    006 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,0,
0,0,0,0, 0,0,0,0,  0,0,1,0;
    007 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,0,
0,0,0,0, 0,0,0,0,  0,0,1,1;
    008 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,0,
0,0,0,0, 0,0,0,0,  0,1,0,0;
    009 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,0,
0,0,0,0, 0,0,0,0,  0,1,0,1;
    010 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,0,
0,0,0,0, 0,0,0,0,  0,1,1,0;
    011 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,0,
0,0,0,0, 0,0,0,0,  0,0,0,1;
    012 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,0,
0,0,0,0, 0,0,0,0,  0,0,1,0;
    013 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,0,
0,0,0,0, 0,0,0,0,  0,0,1,1;
    014 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,0,
0,0,0,0, 0,0,0,0,  0,1,0,0;
    015 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,0,1,
0,0,0,0, 0,0,0,0,  0,0,0,1;
```

```
    016 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,0,1,
0,0,0,0, 0,0,0,0,  0,0,1,0;
    017 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,0,1,
0,0,0,0, 0,0,0,0,  0,0,1,1;
    018 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,0,1,
0,0,0,0, 0,0,0,0,  0,1,0,0;
    019 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,1,
0,0,0,0, 0,0,0,0,  0,0,0,1;
    020 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,1,
0,0,0,0, 0,0,0,0,  0,0,1,0;
    021 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,1,
0,0,0,0, 0,0,0,0,  0,0,1,1;
    022 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,1,1,
0,0,0,0, 0,0,0,0,  0,1,0,0;
    023 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,0,1,
0,0,0,0, 0,0,0,0,  0,0,0,1;
    024 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,0,1,
0,0,0,0, 0,0,0,0,  0,0,1,0;
    025 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,0,0,0,
0,0,0,0, 0,0,0,0,  0,0,0,1;
    026 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 0,0,0,0,
0,0,0,0, 0,0,0,0,  0,0,1,0;
    027 1,1,1,1, 1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1,
1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 1,1,1,1,  1,1,1,1, 0,0,0,0,
0,0,0,0, 0,0,0,0,  0,0,1,1;
    ];


%   Node #  X Axis     Y Axis
Node_Location = [
    001     5          -22.5;
    002     5          -12.5;
    003     12.5          -5;
    004     22.5          -5;
    005     5          -32.5;
    006     -5         -22.5;
    007     -5         -12.5;
    008     12.5           5;
    009     12.5           5;
    010     32.5          -5;
    011     -5         -32.5;
    012     -12.5         -5;
    013     5           12.5;
    014     32.5           5;
    015     -22.5         -5;
    016     -12.5          5;
    017     -5          12.5;
    018     5           22.5;
    019     -32.5         -5;
    020     -22.5          5;
    021     -5          22.5;
    022     5           32.5;
    023     -32.5          5;
    024     -5          32.5;
    025     12.5       -17.5;
    026     17.5       -12.5;
    027     17.5       -17.5;

];

%   Node#   Status   Stat Time chg  Errors?     Seq #
Node_Status = [
    1       0           0               0       0;
    2       0           0               0       0;
    3       0           0               0       0;
    4       0           0               0       0;
    5       0           0               0       0;
    6       0           0               0       0;
    7       0           0               0       0;
    8       0           0               0       0;
    9       0           0               0       0;
    10      0           0               0       0;
    11      0           0               0       0;
    12      0           0               0       0;
    13      0           0               0       0;
    14      0           0               0       0;
    15      0           0               0       0;
    16      0           0               0       0;
    17      0           0               0       0;
    18      0           0               0       0;
    19      0           0               0       0;
    20      0           0               0       0;
    21      0           0               0       0;
    22      0           0               0       0;
    23      0           0               0       0;
    24      0           0               0       0;
    25      0           0               0       0;
    26      0           0               0       0;
    27      2           0               0       0;
    ];


% Node    Time
Wait_Time = [
    1       0;
    2       0;
    3       0;
    4       0;
    5       0;
    6       0;
    7       0;
    8       0;
    9       0;
    10      0;
    11      0;
    12      0;
    13      0;
    14      0;
    15      0;
    16      0;
    17      0;
    18      0;
    19      0;
    20      0;
    21      0;
    22      0;
    23      0;
    24      0;
    25      0;
    26      0;
    27      0;
    ];

%Node       Time    Action from     Action to
Node_Status_Table = [
    0       0      0      0;
    ];

%random keys generated for each node to be shared with the MS
for n = 1:27
    Keys(n,1:128) = randi([0,1],1,128);
end

end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create_Routing_Table_To_Master %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Create_Routing_Table_To_Master()

% Below is the Routing table to be used
global Routing_Table;
%Catergory 1=Send Pri   2=Send Sec
% Node#     Category    Next Hop
Routing_Table = [
    1       1           25;
    2       1           25;
    3       1           26;
    4       1           26;
    5       1           1;
    6       1           1;
    7       1           2;
    8       1           3;
    9       1           4;
    10      1           4;
    11      1           5;
    12      1           7;
    13      1           8;
    14      1           10;
    15      1           12;
    16      1           12;
    17      1           13;
    18      1           13;
    19      1           15;
    20      1           15;
    21      1           18;
    22      1           18;
    23      1           19;
    24      1           22;
    25      1           27;
    26      1           27;
    1       2           2;
    2       2           3;
    3       2           2;
    4       2           3;
    5       2           11;
    6       2           7;
    7       2           6;
    8       2           9;
    9       2           8;
    10      2           14;
    11      2           6;
    12      2           16;
    13      2           17;
    14      2           9;
    15      2           20;
    16      2           17;
    17      2           16;
    18      2           21;
    19      2           23;
    20      2           16;
    21      2           17;
    22      2           24;
    23      2           20;
    24      2           21;
    25      2           27;
    26      2           27;
    ];
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create_Packet_Table %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Create_Packet_Table()
% Create the table to store open packets in

global Open_Packets;
global Starter_Packet;
global Packet;
global Node_Last_Packet;
global Packet_Retrans_Tracker;
global Retrans;
global Initial_Payload;
global Packets;

Packets = zeros(1,3);

Retrans = 0;

% Initialize Packet variable
Packet(1,1:1016) = zeros(1,1016);

% Initialize Packet Retransmit Tracker
% Packet number   NewPack# Trans-Node Timer
Packet_Retrans_Tracker(1,1:4)= [0 0 0 0];

%Packet#   timer   event   # of trans     TransNode RecNode     Errors
Open_Packets = [
    0       0       0           0           0           0        99
    ];

% Frame Control
Starter_Packet(1:16) = [0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0];

% Source MAC Address
Starter_Packet(17:80) = ones(1,64);

%Destination MAC Address
Starter_Packet(81:144) = ones(1,64);

%LOWPAN_IPHC
Starter_Packet(145:160)= randi([0,1],1,16);

% Path/Hop Limit
Starter_Packet(161:162) = [0,0];     % Path
Starter_Packet(163:168) = ones(1,6); % Hop Limit

% Iniialization Vector
% Source Address
    Starter_Packet(169:184) = ones(1,16);

% Destination Address
```

```
    Starter_Packet(185:200) = ones(1,16);

% LOWPAN NHC
    Starter_Packet(201:216) =  randi([0,1],1,16);

%Flags
    Starter_Packet(217:224) =  randi([0,1],1,8);

%Sequence Number
    Starter_Packet(225:256) = ones(1,32);

% Source Port
    Starter_Packet(257:272) =  randi([0,1],1,16);

% Destination Port
    Starter_Packet(273:288) =  randi([0,1],1,16);

% Length
    Starter_Packet(289:296) =  randi([0,1],1,8);

% Payload
aa = 297;
bb = aa + 7;
for n = 1:16
    Starter_Packet(aa:bb) = [1 0 0 1 1 0 0 1];
    aa = aa + 8;
    bb = aa + 7;
end

Initial_Payload(1:128) = Starter_Packet(297:424);

% Padding
Starter_Packet(425:864) = randi([0,1],1,440);

% Message Integrity Code
Starter_Packet(865:992) = ones(1,128);

% Next Header
Starter_Packet(993:1000) = ones(1,8);

% Field Check Sum (CRC)
Starter_Packet(1001:1016) = ones(1,16);

for n = 1:24
    Node_Last_Packet(n,:) = Starter_Packet;
end

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create_Energy_Table %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Create_Energy_Table()
% To determine the amount of energy use from receiving a packet

global Energy_Table;

%   TransNode   Affected Node
Energy_Table = [
    1           2;
    1           5;
    1           6;
    1           25;
    2           7;
    2           3;
    2           1;
    2           25;
    3           8;
    3           4;
    3           2;
    3           26;
    4           9;
    4           10;
    4           3;
    4           26;
    5           1;
    5           11;
    6           11;
    6           1;
    6           7;
    7           6;
    7           2;
    7           12;
    8           13;
    8           3;
    8           9;
    9           8;
    9           4;
    9           14;
    10          14;
    10          4;
    11          5;
    11          6;
    12          7;
    12          15;
    12          16;
    13          8;
    13          17;
    13          18;
    14          10;
    14          9;
    15          12;
    15          19;
    15          20;
    16          17;
    16          12;
    16          20;
    17          16;
    17          13;
    17          21;
    18          13;
    18          21;
    18          22;
    19          23;
    19          15;
    20          15;
    20          16;
    20          23;
    21          17;
    21          18;
    21          24;
    22          18;
    22          24;
    23          19;
    23          20;
    24          21;
    24          22;
    25          1;
    25          2;
    25          26;
    25          27;
    26          3;
    26          4;
    26          25;
    26          27;
    27          25;
    27          26;
];

end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create_Energy_Use %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Create_Energy_Use()
% Keeps track of the amount of energy used per node during the
simulation

global Energy_Use;
% Node  Wakeup  Process   Trans   Rec     Waiting  Go2Slp Post-T-Wait
Sleep
Energy_Use = [
    1     0         0       0       0       0       0       0       0;
    2     0         0       0       0       0       0       0       0;
    3     0         0       0       0       0       0       0       0;
    4     0         0       0       0       0       0       0       0;
    5     0         0       0       0       0       0       0       0;
    6     0         0       0       0       0       0       0       0;
    7     0         0       0       0       0       0       0       0;
    8     0         0       0       0       0       0       0       0;
    9     0         0       0       0       0       0       0       0;
    10    0         0       0       0       0       0       0       0;
    11    0         0       0       0       0       0       0       0;
    12    0         0       0       0       0       0       0       0;
    13    0         0       0       0       0       0       0       0;
    14    0         0       0       0       0       0       0       0;
    15    0         0       0       0       0       0       0       0;
    16    0         0       0       0       0       0       0       0;
    17    0         0       0       0       0       0       0       0;
    18    0         0       0       0       0       0       0       0;
    19    0         0       0       0       0       0       0       0;
    20    0         0       0       0       0       0       0       0;
    21    0         0       0       0       0       0       0       0;
    22    0         0       0       0       0       0       0       0;
    23    0         0       0       0       0       0       0       0;
    24    0         0       0       0       0       0       0       0;
    25    0         0       0       0       0       0       0       0;
    26    0         0       0       0       0       0       0       0;
    27    0         0       0       0       0       0       0       0;
    ];
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create_Event_Table %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Create_Event_Table()
% This is what initiates the events. Events are put in to create the
% sensing events that are simulated within the program.

global Events;
global Orig_Events;
global Scenario_1;
global Scenario_2;
global Scenario_3;
global Scenario_4;
global fileID;
global Enc;

%Choose which one you would like but comment out the others
Event_Table();

% or build your own setup

% %Scenario selection
global Selection;
```

```
% which events to upload per the user's determination in "main" file
if Selection == 1
    Events = Scenario_1;
elseif Selection == 2
    Events = Scenario_2;
elseif Selection == 3
    Events = Scenario_3;
elseif Selection == 4
    Events = Scenario_4;
end

% logs for the events
[a,b] = size(Events);
Orig_Events = a;
Enc = zeros(1,26);
for n = 1:a
    Enc(Events(n,2)) = Enc(Events(n,2)) + 1;
end

fprintf('A total of %i detections were orignally uploaded. \n',a);
fprintf(fileID,'A total of %i detections were orignally uploaded.
\n',a);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Event_Table %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Event_Table()
% This is what initiates the events. Events are put in to create the
% sensing events that are simulated within the program.

%Events used in this thesis, 1-4, preloaded in their individual files
Scenario_11();
Scenario_22();
Scenario_33();
Scenario_44();


End



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Scenario 11 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Scenario_11()

global Scenario_1;
```

```
%time   originating node        destination node
Scenario_1 = [
0       14  27
0       19  27
447     10  27
447     23  27
894     9   27
894     15  27
1342    4   27
1342    20  27
1789    8   27
1789    12  27
2000    14  27
2000    19  27
2236    3   27
2236    16  27
4000    5   27
4000    24  27
4025    3   27
4025    16  27
4447    11  27
4447    22  27
4472    8   27
4472    12  27
4894    1   27
4894    21  27
4919    4   27
4919    20  27
5342    6   27
5342    18  27
5367    9   27
5367    15  27
5789    2   27
5789    17  27
5814    10  27
5814    23  27
6236    7   27
6236    13  27
6261    14  27
6261    19  27
7000    14  27
7000    19  27
7447    10  27
7447    23  27
7894    9   27
7894    15  27
8025    7   27
8025    13  27
8342    4   27
8342    20  27
8472    2   27
8472    17  27
8789    8   27
8789    12  27
8919    6   27
```

8919	18	27
9000	14	27
9000	19	27
9194	9	27
9194	15	27
9236	3	27
9236	16	27
9367	1	27
9367	21	27
9447	10	27
9447	23	27
9814	11	27
9814	22	27
9894	9	27
9894	15	27
10261	5	27
10261	24	27
10342	4	27
10342	20	27
10789	8	27
10789	12	27
11025	3	27
11025	16	27
11236	3	27
11236	16	27
11472	8	27
11472	12	27
11919	4	27
11919	20	27
12367	9	27
12367	15	27
12814	10	27
12814	23	27
13000	5	27
13000	24	27
13025	3	27
13025	16	27
13261	14	27
13261	19	27
13447	11	27
13447	22	27
13472	8	27
13472	12	27
13894	1	27
13894	21	27
13919	4	27
13919	20	27
14342	6	27
14342	18	27
14367	9	27
14367	15	27
14789	2	27
14789	17	27
14814	10	27
14814	23	27

15000	5	27
15000	24	27
15236	7	27
15236	13	27
15261	14	27
15261	19	27
15447	11	27
15447	22	27
15894	1	27
15894	21	27
16342	6	27
16342	18	27
16388	8	27
16388	12	27
16789	2	27
16789	17	27
17025	7	27
17025	13	27
17236	7	27
17236	13	27
17472	2	27
17472	17	27
17919	6	27
17919	18	27
18367	1	27
18367	21	27
18814	11	27
18814	22	27
19000	14	27
19000	19	27
19025	7	27
19025	13	27
19261	5	27
19261	24	27
19447	10	27
19447	23	27
19472	2	27
19472	17	27
19894	9	27
19894	15	27
19919	6	27
19919	18	27
20000	5	27
20000	24	27
20342	4	27
20342	20	27
20367	1	27
20367	21	27
20789	8	27
20789	12	27
20814	11	27
20814	22	27
21236	3	27
21236	16	27
21261	5	27

21261	24	27
22000	14	27
22000	19	27
22447	10	27
22447	23	27
22894	9	27
22894	15	27
23025	3	27
23025	16	27
23342	4	27
23342	20	27
23472	8	27
23472	12	27
23789	8	27
23789	12	27
23919	4	27
23919	20	27
24236	3	27
24236	16	27
24367	9	27
24367	15	27
24814	10	27
24814	23	27
25000	14	27
25000	19	27
25261	14	27
25261	19	27
25447	10	27
25447	23	27
25894	9	27
25894	15	27
26025	3	27
26025	16	27
26342	4	27
26342	20	27
26472	8	27
26472	12	27
26789	8	27
26789	12	27
26919	4	27
26919	20	27
27194	1	27
27194	21	27
27236	3	27
27236	16	27
27367	9	27
27367	15	27
27814	10	27
27814	23	27
28261	14	27
28261	19	27
29025	3	27
29025	16	27
29472	8	27
29472	12	27

29919	4	27
29919	20	27
30000	5	27
30000	24	27
30367	9	27
30367	15	27
30447	11	27
30447	22	27
30814	10	27
30814	23	27
30894	1	27
30894	21	27
31261	14	27
31261	19	27
31342	6	27
31342	18	27
31789	2	27
31789	17	27
32000	5	27
32000	24	27
32236	7	27
32236	13	27
32447	11	27
32447	22	27
32894	1	27
32894	21	27
33342	6	27
33342	18	27
33789	2	27
33789	17	27
34000	5	27
34000	24	27
34025	7	27
34025	13	27
34236	7	27
34236	13	27
34374	3	27
34374	16	27
34388	2	27
34388	17	27
34447	11	27
34447	22	27
34472	2	27
34472	17	27
34894	1	27
34894	21	27
34919	6	27
34919	18	27
35342	6	27
35342	18	27
35367	1	27
35367	21	27
35789	2	27
35789	17	27
35814	11	27

35814	22	27
36025	7	27
36025	13	27
36236	7	27
36236	13	27
36261	5	27
36261	24	27
36472	2	27
36472	17	27
36919	6	27
36919	18	27
37367	1	27
37367	21	27
37814	11	27
37814	22	27
38025	7	27
38025	13	27
38261	5	27
38261	24	27
38472	2	27
38472	17	27
38919	6	27
38919	18	27
39367	1	27
39367	21	27
39814	11	27
39814	22	27
40000	14	27
40000	19	27
40261	5	27
40261	24	27
40447	10	27
40447	23	27
40894	9	27
40894	15	27
41342	4	27
41342	20	27
41568	4	27
41568	20	27
41789	8	27
41789	12	27
42000	14	27
42000	19	27
42236	3	27
42236	16	27
42447	10	27
42447	23	27
42894	9	27
42894	15	27
43342	4	27
43342	20	27
43789	8	27
43789	12	27
44000	14	27
44000	19	27

44025	3	27
44025	16	27
44236	3	27
44236	16	27
44447	10	27
44447	23	27
44472	8	27
44472	12	27
44894	9	27
44894	15	27
44919	4	27
44919	20	27
45342	4	27
45342	20	27
45367	9	27
45367	15	27
45789	8	27
45789	12	27
45814	10	27
45814	23	27
46000	14	27
46000	19	27
46025	3	27
46025	16	27
46236	3	27
46236	16	27
46261	14	27
46261	19	27
46447	10	27
46447	23	27
46472	8	27
46472	12	27
46894	9	27
46894	15	27
46919	4	27
46919	20	27
47342	4	27
47342	20	27
47367	9	27
47367	15	27
47789	8	27
47789	12	27
47814	10	27
47814	23	27
48025	3	27
48025	16	27
48236	3	27
48236	16	27
48261	14	27
48261	19	27
48472	8	27
48472	12	27
48763	10	27
48763	23	27
48919	4	27

```
48919   20  27
49367   9   27
49367   15  27
49814   10  27
49814   23  27
50025   3   27
50025   16  27
50261   14  27
50261   19  27
50472   8   27
50472   12  27
50919   4   27
50919   20  27
51367   9   27
51367   15  27
51814   10  27
51814   23  27
52000   5   27
52000   24  27
52261   14  27
52261   19  27
52374   7   27
52374   13  27
52447   11  27
52447   22  27
52894   1   27
52894   21  27
53342   6   27
53342   18  27
53789   2   27
53789   17  27
54000   5   27
54000   24  27
54236   7   27
54236   13  27
54447   11  27
54447   22  27
54894   1   27
54894   21  27
55342   6   27
55342   18  27
55789   2   27
55789   17  27
56000   5   27
56000   24  27
56025   7   27
56025   13  27
56236   7   27
56236   13  27
56447   11  27
56447   22  27
56472   2   27
56472   17  27
56894   1   27
56894   21  27
```

```
56919   6   27
56919   18  27
57342   6   27
57342   18  27
57367   1   27
57367   21  27
57789   2   27
57789   17  27
57814   11  27
57814   22  27
58000   5   27
58000   24  27
58025   7   27
58025   13  27
58236   7   27
58236   13  27
58261   5   27
58261   24  27
58447   11  27
58447   22  27
58472   2   27
58472   17  27
58894   1   27
58894   21  27
58919   6   27
58919   18  27
59342   6   27
59342   18  27
59367   1   27
59367   21  27
59568   6   27
59568   18  27
59789   2   27
59789   17  27
59814   11  27
59814   22  27
60025   7   27
60025   13  27
60236   7   27
60236   13  27
60261   5   27
60261   24  27
60472   2   27
60472   17  27
60919   6   27
60919   18  27
61367   1   27
61367   21  27
61814   11  27
61814   22  27
62025   7   27
62025   13  27
62261   5   27
62261   24  27
62472   2   27
```

62472	17	27
62919	6	27
62919	18	27
63367	1	27
63367	21	27
63814	11	27
63814	22	27
64261	5	27
64261	24	27
65000	14	27
65000	19	27
65447	10	27
65447	23	27
65894	9	27
65894	15	27
66342	4	27
66342	20	27
66763	11	27
66763	22	27
66789	8	27
66789	12	27
67236	3	27
67236	16	27
68000	14	27
68000	19	27
68447	10	27
68447	23	27
68894	9	27
68894	15	27
69025	3	27
69025	16	27
69342	4	27
69342	20	27
69472	8	27
69472	12	27
69789	8	27
69789	12	27
69919	4	27
69919	20	27
70236	3	27
70236	16	27
70367	15	27
70367	9	27
70814	10	27
70814	23	27
71000	14	27
71000	19	27
71261	14	27
71261	19	27
71447	10	27
71447	23	27
71894	9	27
71894	15	27
72025	3	27
72025	16	27

72342	4	27
72342	20	27
72472	8	27
72472	12	27
72789	8	27
72789	12	27
72919	4	27
72919	20	27
73236	3	27
73236	16	27
73367	9	27
73367	15	27
73814	10	27
73814	23	27
74000	14	27
74000	19	27
74261	14	27
74261	19	27
74447	10	27
74447	23	27
74894	9	27
74894	15	27
75025	3	27
75025	16	27
75342	4	27
75342	20	27
75472	8	27
75472	12	27
75789	8	27
75789	12	27
75919	4	27
75919	20	27
76236	3	27
76236	16	27
76367	9	27
76367	15	27
76814	10	27
76814	23	27
77000	14	27
77000	19	27
77261	14	27
77261	19	27
77447	10	27
77447	23	27
77894	9	27
77894	15	27
78025	3	27
78025	16	27
78342	4	27
78342	20	27
78472	8	27
78472	12	27
78789	8	27
78789	12	27
78919	4	27

78919	20	27
79236	3	27
79236	16	27
79367	9	27
79367	15	27
79814	10	27
79814	23	27
80261	14	27
80261	19	27
81025	3	27
81025	16	27
81472	8	27
81472	12	27
81919	4	27
81919	20	27
82367	9	27
82367	15	27
82814	10	27
82814	23	27
83261	14	27
83261	19	27
85000	5	27
85000	24	27
85447	11	27
85447	22	27
85894	1	27
85894	21	27
86342	6	27
86342	18	27
86789	2	27
86789	17	27
87236	7	27
87236	13	27
88000	5	27
88000	24	27
88447	11	27
88447	22	27
88894	1	27
88894	21	27
89025	7	27
89025	13	27
89342	6	27
89342	18	27
89472	2	27
89472	17	27
89789	2	27
89789	17	27
89919	6	27
89919	18	27
90236	7	27
90236	13	27
90367	1	27
90367	21	27
90814	11	27
90814	22	27

```
91000   5   27
91000   24  27
91261   5   27
91261   24  27
91447   11  27
91447   22  27
91894   1   27
91894   21  27
92025   7   27
92025   13  27
92342   6   27
92342   18  27
92472   2   27
92472   17  27
92789   2   27
92789   17  27
92919   6   27
92919   18  27
93236   7   27
93236   13  27
93367   1   27
93367   21  27
93814   11  27
93814   22  27
94000   5   27
94000   24  27
94261   5   27
94261   24  27
94447   11  27
94447   22  27
94894   1   27
94894   21  27
95025   7   27
95025   13  27
95342   6   27
95342   18  27
95472   2   27
95472   17  27
95789   2   27
95789   17  27
95919   6   27
95919   18  27
96236   7   27
96236   13  27
96367   1   27
96367   21  27
96814   11  27
96814   22  27
97000   5   27
97000   24  27
97261   5   27
97261   24  27
97447   11  27
97447   22  27
97894   1   27
```

```
97894   21  27
98025   7   27
98025   13  27
98342   6   27
98342   18  27
98472   2   27
98472   17  27
98789   2   27
98789   17  27
98919   6   27
98919   18  27
99236   7   27
99236   13  27
99367   1   27
99367   21  27
99814   11  27
99814   22  27
100261  5   27
100261  24  27
101025  7   27
101025  13  27
101472  2   27
101472  17  27
101919  6   27
101919  18  27
102000  14  27
102000  19  27
102367  1   27
102367  21  27
102447  10  27
102447  23  27
102814  11  27
102814  22  27
102894  9   27
102894  15  27
103261  5   27
103261  24  27
103342  4   27
103342  20  27
103789  8   27
103789  12  27
104236  3   27
104236  16  27
105000  14  27
105000  14  27
105000  19  27
105000  19  27
105447  10  27
105447  23  27
105894  9   27
105894  15  27
106000  14  27
106000  19  27
106025  3   27
106025  16  27
```

106342	4	27
106342	20	27
106472	8	27
106472	12	27
106789	8	27
106789	12	27
106919	4	27
106919	20	27
107236	3	27
107236	16	27
107367	9	27
107367	15	27
107814	10	27
107814	23	27
108000	14	27
108000	19	27
108261	14	27
108261	19	27
108447	10	27
108447	23	27
108894	9	27
108894	15	27
109025	3	27
109025	16	27
109342	4	27
109342	20	27
109472	8	27
109472	12	27
109789	8	27
109789	12	27
109919	4	27
109919	20	27
110236	3	27
110236	16	27
110367	9	27
110367	15	27
110814	10	27
110814	23	27
111000	14	27
111000	19	27
111261	14	27
111261	19	27
111447	10	27
111447	23	27
111894	9	27
111894	15	27
112000	5	27
112000	24	27
112025	3	27
112025	16	27
112194	9	27
112194	15	27
112342	4	27
112342	20	27
112472	8	27

```
112472  12  27
112789  8   27
112789  12  27
112919  4   27
112919  20  27
113000  5   27
113000  24  27
113194  9   27
113194  15  27
113236  3   27
113236  16  27
113367  9   27
113367  15  27
113814  10  27
113814  23  27
114000  14  27
114000  19  27
114261  14  27
114261  19  27
114447  10  27
114447  23  27
114894  9   27
114894  15  27
115025  3   27
115025  16  27
115342  4   27
115342  20  27
115472  8   27
115472  12  27
115789  8   27
115789  12  27
115919  4   27
115919  20  27
116236  3   27
116236  16  27
116367  9   27
116367  15  27
116814  10  27
116814  23  27
117000  14  27
117000  19  27
117261  14  27
117261  19  27
117447  10  27
117447  23  27
117894  9   27
117894  15  27
118025  3   27
118025  16  27
118342  4   27
118342  20  27
118472  8   27
118472  12  27
118789  8   27
118789  12  27
```

118919	4	27
118919	20	27
119194	1	27
119194	21	27
119236	3	27
119236	16	27
119367	9	27
119367	15	27
119388	8	27
119388	12	27
119814	10	27
119814	23	27
120194	1	27
120194	21	27
120261	14	27
120261	19	27
120388	8	27
120388	12	27
121025	3	27
121025	16	27
121472	8	27
121472	12	27
121919	4	27
121919	20	27
122000	5	27
122000	24	27
122367	9	27
122367	15	27
122447	11	27
122447	22	27
122814	10	27
122814	23	27
122894	1	27
122894	21	27
123261	14	27
123261	19	27
123342	6	27
123342	18	27
123789	2	27
123789	17	27
124236	7	27
124236	13	27
125000	5	27
125000	24	27
125447	11	27
125447	22	27
125894	1	27
125894	21	27
126025	7	27
126025	13	27
126342	6	27
126342	18	27
126388	2	27
126388	17	27
126472	2	27

```
126472  17  27
126789  2   27
126789  17  27
126919  6   27
126919  18  27
127236  7   27
127236  13  27
127367  1   27
127367  21  27
127388  2   27
127388  17  27
127814  11  27
127814  22  27
128000  5   27
128000  24  27
128261  5   27
128261  24  27
128447  11  27
128447  22  27
128894  1   27
128894  21  27
129025  7   27
129025  13  27
129342  6   27
129342  18  27
129472  2   27
129472  17  27
129789  2   27
129789  17  27
129919  6   27
129919  18  27
130236  7   27
130236  13  27
130367  1   27
130367  21  27
130814  11  27
130814  22  27
131000  5   27
131000  24  27
131261  5   27
131261  24  27
131447  11  27
131447  22  27
131894  1   27
131894  21  27
132025  7   27
132025  13  27
132342  6   27
132342  18  27
132472  2   27
132472  17  27
132789  2   27
132789  17  27
132919  6   27
132919  18  27
```

133236	7	27
133236	13	27
133367	1	27
133367	21	27
133814	11	27
133814	22	27
134000	5	27
134000	24	27
134261	5	27
134261	24	27
134447	11	27
134447	22	27
134894	1	27
134894	21	27
135025	7	27
135025	13	27
135342	6	27
135342	18	27
135472	2	27
135472	17	27
135789	2	27
135789	17	27
135919	6	27
135919	18	27
136236	7	27
136236	13	27
136367	1	27
136367	21	27
136814	11	27
136814	22	27
137000	5	27
137000	24	27
137261	5	27
137261	24	27
137374	3	27
137374	16	27
137447	11	27
137447	22	27
137894	1	27
137894	21	27
138025	7	27
138025	13	27
138342	6	27
138342	18	27
138374	3	27
138374	16	27
138472	2	27
138472	17	27
138789	2	27
138789	17	27
138919	6	27
138919	18	27
139236	7	27
139236	13	27
139367	1	27

```
139367  21  27
139814  11  27
139814  22  27
140261  5   27
140261  24  27
141025  7   27
141025  13  27
141472  2   27
141472  17  27
141919  6   27
141919  18  27
142367  1   27
142367  21  27
142814  11  27
142814  22  27
143261  5   27
143261  24  27
144374  7   27
144374  13  27
144568  4   27
144568  20  27
145374  7   27
145374  13  27
145568  4   27
145568  20  27
151568  6   27
151568  18  27
151763  10  27
151763  23  27
152568  6   27
152568  18  27
152763  10  27
152763  23  27
158763  11  27
158763  22  27
159763  11  27
159763  22  27
];

end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Scenario_22 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Scenario_22()

global Scenario_2;

Scenario_2 = [
0       14  27
0       19  27
```

319	10	27
319	23	27
639	9	27
639	15	27
958	4	27
958	20	27
1278	8	27
1278	12	27
1597	3	27
1597	16	27
2875	3	27
2875	16	27
3195	8	27
3195	12	27
3514	4	27
3514	20	27
3834	9	27
3834	15	27
4153	10	27
4153	23	27
4473	14	27
4473	19	27
5000	5	27
5000	24	27
5319	11	27
5319	22	27
5639	1	27
5639	21	27
5958	6	27
5958	18	27
6278	2	27
6278	17	27
6597	7	27
6597	13	27
7875	7	27
7875	13	27
8195	2	27
8195	17	27
8514	6	27
8514	18	27
8834	1	27
8834	21	27
9153	11	27
9153	22	27
9473	5	27
9473	24	27
10000	14	27
10000	19	27
10319	10	27
10319	23	27
10639	9	27
10639	15	27
10958	4	27
10958	20	27
11278	8	27

```
11278   12  27
11597   3   27
11597   16  27
12000   14  27
12000   19  27
12319   10  27
12319   23  27
12639   9   27
12639   15  27
12875   3   27
12875   16  27
12958   4   27
12958   20  27
13195   8   27
13195   12  27
13278   8   27
13278   12  27
13514   4   27
13514   20  27
13597   3   27
13597   16  27
13834   9   27
13834   15  27
14153   10  27
14153   23  27
14473   14  27
14473   19  27
14875   3   27
14875   16  27
15195   8   27
15195   12  27
15514   4   27
15514   20  27
15834   9   27
15834   15  27
16153   10  27
16153   23  27
16473   14  27
16473   19  27
17000   5   27
17000   24  27
17319   11  27
17319   22  27
17639   1   27
17639   21  27
17958   6   27
17958   18  27
18278   2   27
18278   17  27
18597   7   27
18597   13  27
19000   5   27
19000   24  27
19319   11  27
19319   22  27
```

```
19639   1   27
19639   21  27
19875   7   27
19875   13  27
19958   6   27
19958   18  27
20195   2   27
20195   17  27
20278   2   27
20278   17  27
20514   6   27
20514   18  27
20597   7   27
20597   13  27
20834   1   27
20834   21  27
21153   11  27
21153   22  27
21473   5   27
21473   24  27
21875   7   27
21875   13  27
22195   2   27
22195   17  27
22514   6   27
22514   18  27
22834   1   27
22834   21  27
23000   14  27
23000   19  27
23153   11  27
23153   22  27
23319   10  27
23319   23  27
23473   5   27
23473   24  27
23639   9   27
23639   15  27
23958   4   27
23958   20  27
24278   8   27
24278   12  27
24597   3   27
24597   16  27
25000   14  27
25000   19  27
25319   10  27
25319   23  27
25639   9   27
25639   15  27
25875   3   27
25875   16  27
25958   4   27
25958   20  27
26195   8   27
```

26195	12	27
26278	8	27
26278	12	27
26514	4	27
26514	20	27
26597	3	27
26597	16	27
26834	9	27
26834	15	27
27000	14	27
27000	19	27
27153	10	27
27153	23	27
27319	10	27
27319	23	27
27473	14	27
27473	19	27
27639	9	27
27639	15	27
27875	3	27
27875	16	27
27958	4	27
27958	20	27
28195	8	27
28195	12	27
28278	8	27
28278	12	27
28514	4	27
28514	20	27
28597	3	27
28597	16	27
28834	9	27
28834	15	27
29153	10	27
29153	23	27
29473	14	27
29473	19	27
29875	3	27
29875	16	27
30195	8	27
30195	12	27
30514	4	27
30514	20	27
30834	9	27
30834	15	27
31000	5	27
31000	24	27
31153	10	27
31153	23	27
31319	11	27
31319	22	27
31473	14	27
31473	19	27
31639	1	27
31639	21	27

31958	6	27
31958	18	27
32278	2	27
32278	17	27
32597	7	27
32597	13	27
33000	5	27
33000	24	27
33319	11	27
33319	22	27
33639	1	27
33639	21	27
33875	7	27
33875	13	27
33958	6	27
33958	18	27
34195	2	27
34195	17	27
34278	2	27
34278	17	27
34514	6	27
34514	18	27
34597	7	27
34597	13	27
34834	1	27
34834	21	27
35000	5	27
35000	24	27
35153	11	27
35153	22	27
35319	11	27
35319	22	27
35473	5	27
35473	24	27
35639	1	27
35639	21	27
35875	7	27
35875	13	27
35958	6	27
35958	18	27
36195	2	27
36195	17	27
36278	2	27
36278	17	27
36514	6	27
36514	18	27
36597	7	27
36597	13	27
36834	1	27
36834	21	27
37153	11	27
37153	22	27
37473	5	27
37473	24	27
37875	7	27

37875	13	27
38195	2	27
38195	17	27
38514	6	27
38514	18	27
38834	1	27
38834	21	27
39153	11	27
39153	22	27
39473	5	27
39473	24	27
41000	14	27
41000	19	27
41319	10	27
41319	23	27
41639	9	27
41639	15	27
41958	4	27
41958	20	27
42278	8	27
42278	12	27
42597	3	27
42597	16	27
43000	14	27
43000	19	27
43319	10	27
43319	23	27
43639	9	27
43639	15	27
43875	3	27
43875	16	27
43958	4	27
43958	20	27
44195	8	27
44195	12	27
44278	8	27
44278	12	27
44514	4	27
44514	20	27
44597	3	27
44597	16	27
44834	9	27
44834	15	27
45000	14	27
45000	19	27
45153	10	27
45153	23	27
45319	10	27
45319	23	27
45473	14	27
45473	19	27
45639	9	27
45639	15	27
45875	3	27
45875	16	27

45958	4	27
45958	20	27
46195	8	27
46195	12	27
46278	8	27
46278	12	27
46514	4	27
46514	20	27
46597	3	27
46597	16	27
46834	9	27
46834	15	27
47000	14	27
47000	19	27
47153	10	27
47153	23	27
47319	10	27
47319	23	27
47473	14	27
47473	19	27
47639	9	27
47639	15	27
47875	3	27
47875	16	27
47958	4	27
47958	20	27
48195	8	27
48195	12	27
48278	8	27
48278	12	27
48514	4	27
48514	20	27
48597	3	27
48597	16	27
48834	9	27
48834	15	27
49153	10	27
49153	23	27
49473	14	27
49473	19	27
49875	3	27
49875	16	27
50000	5	27
50000	24	27
50195	8	27
50195	12	27
50319	11	27
50319	22	27
50514	4	27
50514	20	27
50639	1	27
50639	21	27
50834	9	27
50834	15	27
50958	6	27

```
50958   18  27
51153   10  27
51153   23  27
51278   2   27
51278   17  27
51473   14  27
51473   19  27
51597   7   27
51597   13  27
52000   5   27
52000   24  27
52319   11  27
52319   22  27
52639   1   27
52639   21  27
52875   7   27
52875   13  27
52958   6   27
52958   18  27
53195   2   27
53195   17  27
53278   2   27
53278   17  27
53514   6   27
53514   18  27
53597   7   27
53597   13  27
53834   1   27
53834   21  27
54000   5   27
54000   24  27
54153   11  27
54153   22  27
54319   11  27
54319   22  27
54473   5   27
54473   24  27
54639   1   27
54639   21  27
54875   7   27
54875   13  27
54958   6   27
54958   18  27
55195   2   27
55195   17  27
55278   2   27
55278   17  27
55514   6   27
55514   18  27
55597   7   27
55597   13  27
55834   1   27
55834   21  27
56000   5   27
56000   24  27
```

56153	11	27
56153	22	27
56319	11	27
56319	22	27
56473	5	27
56473	24	27
56639	1	27
56639	21	27
56875	7	27
56875	13	27
56958	6	27
56958	18	27
57195	2	27
57195	17	27
57278	2	27
57278	17	27
57514	6	27
57514	18	27
57597	7	27
57597	13	27
57834	1	27
57834	21	27
58153	11	27
58153	22	27
58473	5	27
58473	24	27
58875	7	27
58875	13	27
59195	2	27
59195	17	27
59514	6	27
59514	18	27
59834	1	27
59834	21	27
60000	14	27
60000	14	27
60000	19	27
60000	19	27
60153	11	27
60153	22	27
60319	10	27
60319	23	27
60473	5	27
60473	24	27
60500	14	27
60500	19	27
60639	9	27
60639	15	27
60958	4	27
60958	20	27
61000	19	27
61000	14	27
61278	8	27
61278	12	27
61500	19	27

```
61500   14  27
61597   3   27
61597   16  27
62000   14  27
62000   19  27
62319   10  27
62319   23  27
62639   9   27
62639   15  27
62875   3   27
62875   16  27
62958   4   27
62958   20  27
63195   8   27
63195   12  27
63278   8   27
63278   12  27
63514   4   27
63514   20  27
63597   3   27
63597   16  27
63834   9   27
63834   15  27
64000   14  27
64000   19  27
64153   10  27
64153   23  27
64319   10  27
64319   23  27
64473   14  27
64473   19  27
64639   9   27
64639   15  27
64875   3   27
64875   16  27
64958   4   27
64958   20  27
65195   8   27
65195   12  27
65278   8   27
65278   12  27
65514   4   27
65514   20  27
65597   3   27
65597   16  27
65834   9   27
65834   15  27
66000   14  27
66000   19  27
66153   10  27
66153   23  27
66319   10  27
66319   23  27
66473   14  27
66473   19  27
```

66639	9	27
66639	15	27
66875	3	27
66875	16	27
66958	4	27
66958	20	27
67194	9	27
67194	15	27
67195	8	27
67195	12	27
67278	8	27
67278	12	27
67514	4	27
67514	20	27
67597	3	27
67597	16	27
67694	9	27
67694	15	27
67834	9	27
67834	15	27
68000	14	27
68000	19	27
68153	10	27
68153	23	27
68194	15	27
68194	9	27
68319	10	27
68319	23	27
68473	14	27
68473	19	27
68639	9	27
68639	15	27
68694	15	27
68694	9	27
68875	3	27
68875	16	27
68958	4	27
68958	20	27
69195	8	27
69195	12	27
69278	8	27
69278	12	27
69514	4	27
69514	20	27
69597	3	27
69597	16	27
69834	9	27
69834	15	27
70000	5	27
70000	24	27
70153	10	27
70153	23	27
70473	14	27
70473	19	27
70875	3	27

70875	16	27
71000	5	27
71000	24	27
71195	8	27
71195	12	27
71514	4	27
71514	20	27
71834	9	27
71834	15	27
72000	5	27
72000	24	27
72153	10	27
72153	23	27
72319	11	27
72319	22	27
72473	14	27
72473	19	27
72639	1	27
72639	21	27
72958	6	27
72958	18	27
73278	2	27
73278	17	27
73597	7	27
73597	13	27
74000	5	27
74000	24	27
74319	11	27
74319	22	27
74388	8	27
74388	12	27
74639	1	27
74639	21	27
74875	7	27
74875	13	27
74888	8	27
74888	12	27
74958	6	27
74958	18	27
75195	2	27
75195	17	27
75278	2	27
75278	17	27
75388	12	27
75388	8	27
75514	6	27
75514	18	27
75597	7	27
75597	13	27
75834	1	27
75834	21	27
75888	12	27
75888	8	27
76000	5	27
76000	5	27

76000	24	27
76000	24	27
76153	11	27
76153	22	27
76319	11	27
76319	11	27
76319	22	27
76319	22	27
76473	5	27
76473	24	27
76639	1	27
76639	1	27
76639	21	27
76639	21	27
76875	7	27
76875	13	27
76958	6	27
76958	6	27
76958	18	27
76958	18	27
77194	1	27
77194	21	27
77195	2	27
77195	17	27
77278	2	27
77278	2	27
77278	17	27
77278	17	27
77514	6	27
77514	18	27
77597	7	27
77597	7	27
77597	13	27
77597	13	27
77834	1	27
77834	21	27
78000	5	27
78000	24	27
78153	11	27
78153	22	27
78194	1	27
78194	21	27
78319	11	27
78319	22	27
78473	5	27
78473	24	27
78639	1	27
78639	21	27
78875	7	27
78875	7	27
78875	13	27
78875	13	27
78958	6	27
78958	18	27
79195	2	27

79195	2	27
79195	17	27
79195	17	27
79278	2	27
79278	17	27
79514	6	27
79514	6	27
79514	18	27
79514	18	27
79597	7	27
79597	13	27
79834	1	27
79834	1	27
79834	21	27
79834	21	27
80000	5	27
80000	24	27
80153	11	27
80153	11	27
80153	22	27
80153	22	27
80319	11	27
80319	22	27
80473	5	27
80473	5	27
80473	24	27
80473	24	27
80639	1	27
80639	21	27
80875	7	27
80875	13	27
80958	6	27
80958	18	27
81195	2	27
81195	17	27
81278	2	27
81278	17	27
81514	6	27
81514	18	27
81597	7	27
81597	13	27
81834	1	27
81834	21	27
82153	11	27
82153	22	27
82473	5	27
82473	24	27
82875	7	27
82875	13	27
83195	2	27
83195	17	27
83514	6	27
83514	18	27
83834	1	27
83834	21	27

84153	11	27
84153	22	27
84388	2	27
84388	17	27
84473	5	27
84473	24	27
85000	14	27
85000	19	27
85319	10	27
85319	23	27
85388	2	27
85388	17	27
85639	9	27
85639	15	27
85958	4	27
85958	20	27
86278	8	27
86278	12	27
86597	3	27
86597	16	27
87000	14	27
87000	19	27
87319	10	27
87319	23	27
87639	9	27
87639	15	27
87875	3	27
87875	16	27
87958	4	27
87958	20	27
88195	8	27
88195	12	27
88278	8	27
88278	12	27
88514	4	27
88514	20	27
88597	3	27
88597	16	27
88834	9	27
88834	15	27
89000	14	27
89000	19	27
89153	10	27
89153	23	27
89319	10	27
89319	23	27
89473	14	27
89473	19	27
89639	9	27
89639	15	27
89875	3	27
89875	16	27
89958	4	27
89958	20	27
90195	8	27

90195	12	27
90278	8	27
90278	12	27
90514	4	27
90514	20	27
90597	3	27
90597	16	27
90834	9	27
90834	15	27
91000	14	27
91000	19	27
91153	10	27
91153	23	27
91319	10	27
91319	23	27
91473	14	27
91473	19	27
91639	9	27
91639	15	27
91875	3	27
91875	16	27
91958	4	27
91958	20	27
92195	8	27
92195	12	27
92278	8	27
92278	12	27
92374	3	27
92374	16	27
92514	4	27
92514	20	27
92597	3	27
92597	16	27
92834	9	27
92834	15	27
92874	3	27
92874	16	27
93000	14	27
93000	19	27
93153	10	27
93153	23	27
93319	10	27
93319	23	27
93374	3	27
93374	16	27
93473	14	27
93473	19	27
93639	9	27
93639	15	27
93874	3	27
93874	16	27
93875	3	27
93875	16	27
93958	4	27
93958	20	27

94195	8	27
94195	12	27
94278	8	27
94278	12	27
94514	4	27
94514	20	27
94597	3	27
94597	16	27
94834	9	27
94834	15	27
95000	14	27
95000	19	27
95153	10	27
95153	23	27
95319	10	27
95319	23	27
95473	14	27
95473	19	27
95639	9	27
95639	15	27
95875	3	27
95875	16	27
95958	4	27
95958	20	27
96195	8	27
96195	12	27
96278	8	27
96278	12	27
96514	4	27
96514	20	27
96597	3	27
96597	16	27
96834	9	27
96834	15	27
97153	10	27
97153	23	27
97473	14	27
97473	19	27
97875	3	27
97875	16	27
98195	8	27
98195	12	27
98514	4	27
98514	20	27
98834	9	27
98834	15	27
99153	10	27
99153	23	27
99473	14	27
99473	19	27
99568	4	27
99568	20	27
100000	5	27
100000	24	27
100068	4	27

```
100068  20  27
100319  11  27
100319  22  27
100568  4   27
100568  20  27
100639  1   27
100639  21  27
100958  6   27
100958  18  27
101068  4   27
101068  20  27
101278  2   27
101278  17  27
101597  7   27
101597  13  27
102000  5   27
102000  24  27
102319  11  27
102319  22  27
102374  7   27
102374  13  27
102639  1   27
102639  21  27
102875  7   27
102875  13  27
102958  6   27
102958  18  27
103195  2   27
103195  17  27
103278  2   27
103278  17  27
103374  7   27
103374  13  27
103514  6   27
103514  18  27
103597  7   27
103597  13  27
103834  1   27
103834  21  27
104000  5   27
104000  24  27
104153  11  27
104153  22  27
104319  11  27
104319  22  27
104473  5   27
104473  24  27
104639  1   27
104639  21  27
104875  7   27
104875  13  27
104958  6   27
104958  18  27
105195  2   27
105195  17  27
```

105278	2	27
105278	17	27
105514	6	27
105514	18	27
105597	7	27
105597	13	27
105834	1	27
105834	21	27
106000	5	27
106000	24	27
106153	11	27
106153	22	27
106319	11	27
106319	22	27
106473	5	27
106473	24	27
106639	1	27
106639	21	27
106763	10	27
106763	23	27
106875	7	27
106875	13	27
106958	6	27
106958	18	27
107195	2	27
107195	17	27
107263	10	27
107263	23	27
107278	2	27
107278	17	27
107514	6	27
107514	18	27
107597	7	27
107597	13	27
107763	10	27
107763	23	27
107834	1	27
107834	21	27
108000	5	27
108000	24	27
108153	11	27
108153	22	27
108263	10	27
108263	23	27
108319	11	27
108319	22	27
108473	5	27
108473	24	27
108639	1	27
108639	21	27
108875	7	27
108875	13	27
108958	6	27
108958	18	27
109195	2	27

109195	17	27
109278	2	27
109278	17	27
109514	6	27
109514	18	27
109568	6	27
109568	18	27
109597	7	27
109597	13	27
109834	1	27
109834	21	27
110000	5	27
110000	24	27
110153	11	27
110153	22	27
110319	11	27
110319	22	27
110473	5	27
110473	24	27
110568	6	27
110568	18	27
110639	1	27
110639	21	27
110875	7	27
110875	13	27
110958	6	27
110958	18	27
111195	2	27
111195	17	27
111278	2	27
111278	17	27
111514	6	27
111514	18	27
111597	7	27
111597	13	27
111834	1	27
111834	21	27
112153	11	27
112153	22	27
112473	5	27
112473	24	27
112875	7	27
112875	13	27
113195	2	27
113195	17	27
113514	6	27
113514	18	27
113834	1	27
113834	21	27
114153	11	27
114153	22	27
114473	5	27
114473	24	27
116763	11	27
116763	22	27

```
117763  11  27
117763  22  27
];

end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Scenario_33 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Scenario_33()

global Scenario_3;

%time   originating node        destination node
Scenario_3 = [
0       19  27
0       14  27
249     23  27
249     10  27
497     15  27
497     9   27
746     20  27
746     4   27
994     12  27
994     8   27
1243    16  27
1243    3   27
2000    19  27
2000    14  27
2237    3   27
2237    16  27
2249    23  27
2249    10  27
2485    8   27
2485    12  27
2497    15  27
2497    9   27
2734    4   27
2734    20  27
2746    20  27
2746    4   27
2982    9   27
2982    15  27
2994    12  27
2994    8   27
3231    10  27
3231    23  27
3243    16  27
3243    3   27
3479    14  27
3479    19  27
```

```
4000    19  27
4000    14  27
4237    3   27
4237    16  27
4249    23  27
4249    10  27
4485    8   27
4485    12  27
4497    15  27
4497    9   27
4734    4   27
4734    20  27
4746    20  27
4746    4   27
4982    9   27
4982    15  27
4994    12  27
4994    8   27
5231    10  27
5231    23  27
5243    16  27
5243    3   27
5479    14  27
5479    19  27
6000    19  27
6000    14  27
6237    3   27
6237    16  27
6249    23  27
6249    10  27
6485    8   27
6485    12  27
6497    15  27
6497    9   27
6734    4   27
6734    20  27
6746    20  27
6746    4   27
6982    9   27
6982    15  27
6994    12  27
6994    8   27
7231    10  27
7231    23  27
7243    16  27
7243    3   27
7479    14  27
7479    19  27
8000    19  27
8000    14  27
8237    3   27
8237    16  27
8249    23  27
8249    10  27
8485    8   27
```

8485	12	27
8497	15	27
8497	9	27
8734	4	27
8734	20	27
8746	20	27
8746	4	27
8982	9	27
8982	15	27
8994	12	27
8994	8	27
9231	10	27
9231	23	27
9243	16	27
9243	3	27
9479	14	27
9479	19	27
10237	3	27
10237	16	27
10485	8	27
10485	12	27
10734	4	27
10734	20	27
10982	9	27
10982	15	27
11231	10	27
11231	23	27
11479	14	27
11479	19	27
12000	24	27
12000	5	27
12249	22	27
12249	11	27
12497	21	27
12497	1	27
12746	18	27
12746	6	27
12994	17	27
12994	2	27
13243	13	27
13243	7	27
14000	24	27
14000	5	27
14237	7	27
14237	13	27
14249	22	27
14249	11	27
14485	2	27
14485	17	27
14497	21	27
14497	1	27
14734	6	27
14734	18	27
14746	18	27
14746	6	27

14982	1	27
14982	21	27
14994	17	27
14994	2	27
15231	11	27
15231	22	27
15243	13	27
15243	7	27
15479	5	27
15479	24	27
16000	24	27
16000	5	27
16237	7	27
16237	13	27
16249	22	27
16249	11	27
16485	2	27
16485	17	27
16497	21	27
16497	1	27
16734	6	27
16734	18	27
16746	18	27
16746	6	27
16982	1	27
16982	21	27
16994	17	27
16994	2	27
17231	11	27
17231	22	27
17243	13	27
17243	7	27
17479	5	27
17479	24	27
18000	24	27
18000	5	27
18237	7	27
18237	13	27
18249	22	27
18249	11	27
18485	2	27
18485	17	27
18497	21	27
18497	1	27
18734	6	27
18734	18	27
18746	18	27
18746	6	27
18982	1	27
18982	21	27
18994	17	27
18994	2	27
19231	11	27
19231	22	27
19243	13	27

19243	7	27
19479	5	27
19479	24	27
20000	24	27
20000	5	27
20237	7	27
20237	13	27
20249	22	27
20249	11	27
20485	2	27
20485	17	27
20497	21	27
20497	1	27
20734	6	27
20734	18	27
20746	18	27
20746	6	27
20982	1	27
20982	21	27
20994	17	27
20994	2	27
21231	11	27
21231	22	27
21243	13	27
21243	7	27
21479	5	27
21479	24	27
22237	7	27
22237	13	27
22485	2	27
22485	17	27
22734	6	27
22734	18	27
22982	1	27
22982	21	27
23231	11	27
23231	22	27
23479	5	27
23479	24	27
24000	19	27
24000	14	27
24249	23	27
24249	10	27
24497	15	27
24497	9	27
24746	20	27
24746	4	27
24994	12	27
24994	8	27
25243	16	27
25243	3	27
26000	19	27
26000	14	27
26237	3	27
26237	16	27

26249	23	27
26249	10	27
26485	8	27
26485	12	27
26497	15	27
26497	9	27
26734	4	27
26734	20	27
26746	20	27
26746	4	27
26982	9	27
26982	15	27
26994	12	27
26994	8	27
27231	10	27
27231	23	27
27243	16	27
27243	3	27
27479	14	27
27479	19	27
28000	19	27
28000	14	27
28237	3	27
28237	16	27
28249	23	27
28249	10	27
28485	8	27
28485	12	27
28497	15	27
28497	9	27
28734	4	27
28734	20	27
28746	20	27
28746	4	27
28982	9	27
28982	15	27
28994	12	27
28994	8	27
29231	10	27
29231	23	27
29243	16	27
29243	3	27
29479	14	27
29479	19	27
30000	19	27
30000	14	27
30000	19	27
30000	14	27
30237	3	27
30237	16	27
30249	23	27
30249	10	27
30485	8	27
30485	12	27
30497	15	27

30497	9	27
30734	4	27
30734	20	27
30746	20	27
30746	4	27
30982	9	27
30982	15	27
30994	12	27
30994	8	27
31231	10	27
31231	23	27
31243	16	27
31243	3	27
31479	14	27
31479	19	27
32000	19	27
32000	14	27
32237	3	27
32237	16	27
32249	23	27
32249	10	27
32485	8	27
32485	12	27
32497	15	27
32497	9	27
32734	4	27
32734	20	27
32746	20	27
32746	4	27
32982	9	27
32982	15	27
32994	12	27
32994	8	27
33231	10	27
33231	23	27
33243	16	27
33243	3	27
33479	14	27
33479	19	27
34237	3	27
34237	16	27
34485	8	27
34485	12	27
34734	4	27
34734	20	27
34982	9	27
34982	15	27
35231	10	27
35231	23	27
35479	14	27
35479	19	27
36000	24	27
36000	5	27
36249	22	27
36249	11	27

36497	21	27
36497	1	27
36746	18	27
36746	6	27
36994	17	27
36994	2	27
37194	15	27
37194	9	27
37243	13	27
37243	7	27
38000	24	27
38000	5	27
38237	7	27
38237	13	27
38249	22	27
38249	11	27
38485	2	27
38485	17	27
38497	21	27
38497	1	27
38734	6	27
38734	18	27
38746	18	27
38746	6	27
38982	1	27
38982	21	27
38994	17	27
38994	2	27
39231	11	27
39231	22	27
39243	13	27
39243	7	27
39479	5	27
39479	24	27
40000	24	27
40000	5	27
40237	7	27
40237	13	27
40249	22	27
40249	11	27
40485	2	27
40485	17	27
40497	21	27
40497	1	27
40734	6	27
40734	18	27
40746	18	27
40746	6	27
40982	1	27
40982	21	27
40994	17	27
40994	2	27
41231	11	27
41231	22	27
41243	13	27

41243	7	27
41479	5	27
41479	24	27
42000	24	27
42000	5	27
42237	7	27
42237	13	27
42249	22	27
42249	11	27
42485	2	27
42485	17	27
42497	21	27
42497	1	27
42734	6	27
42734	18	27
42746	18	27
42746	6	27
42982	1	27
42982	21	27
42994	17	27
42994	2	27
43231	11	27
43231	22	27
43243	13	27
43243	7	27
43479	5	27
43479	24	27
44000	24	27
44000	5	27
44237	7	27
44237	13	27
44249	22	27
44249	11	27
44388	12	27
44388	8	27
44485	2	27
44485	17	27
44497	21	27
44497	1	27
44734	6	27
44734	18	27
44746	18	27
44746	6	27
44982	1	27
44982	21	27
44994	17	27
44994	2	27
45231	11	27
45231	22	27
45243	13	27
45243	7	27
45479	5	27
45479	24	27
46237	7	27
46237	13	27

46485	2	27
46485	17	27
46734	6	27
46734	18	27
46982	1	27
46982	21	27
47231	11	27
47231	22	27
47479	5	27
47479	24	27
48000	19	27
48000	14	27
48249	23	27
48249	10	27
48497	15	27
48497	9	27
48746	20	27
48746	4	27
48994	12	27
48994	8	27
49243	16	27
49243	3	27
50000	19	27
50000	14	27
50000	5	27
50000	24	27
50237	3	27
50237	16	27
50249	23	27
50249	10	27
50485	8	27
50485	12	27
50497	15	27
50497	9	27
50734	4	27
50734	20	27
50746	20	27
50746	4	27
50982	9	27
50982	15	27
50994	12	27
50994	8	27
51231	10	27
51231	23	27
51243	16	27
51243	3	27
51479	14	27
51479	19	27
52000	19	27
52000	14	27
52237	3	27
52237	16	27
52249	23	27
52249	10	27
52485	8	27

52485	12	27
52497	15	27
52497	9	27
52734	4	27
52734	20	27
52746	20	27
52746	4	27
52982	9	27
52982	15	27
52994	12	27
52994	8	27
53231	10	27
53231	23	27
53243	16	27
53243	3	27
53479	14	27
53479	19	27
54000	19	27
54000	14	27
54237	3	27
54237	16	27
54249	23	27
54249	10	27
54485	8	27
54485	12	27
54497	15	27
54497	9	27
54734	4	27
54734	20	27
54746	20	27
54746	4	27
54982	9	27
54982	15	27
54994	12	27
54994	8	27
55231	10	27
55231	23	27
55243	16	27
55243	3	27
55479	14	27
55479	19	27
56000	19	27
56000	14	27
56237	3	27
56237	16	27
56249	23	27
56249	10	27
56485	8	27
56485	12	27
56497	15	27
56497	9	27
56734	4	27
56734	20	27
56746	20	27
56746	4	27

56982	9	27
56982	15	27
56994	12	27
56994	8	27
57194	1	27
57194	21	27
57231	10	27
57231	23	27
57243	16	27
57243	3	27
57479	14	27
57479	19	27
58237	3	27
58237	16	27
58485	8	27
58485	12	27
58734	4	27
58734	20	27
58982	9	27
58982	15	27
59231	10	27
59231	23	27
59479	14	27
59479	19	27
60000	24	27
60000	5	27
60249	22	27
60249	11	27
60497	21	27
60497	1	27
60746	18	27
60746	6	27
60994	17	27
60994	2	27
61243	13	27
61243	7	27
62000	24	27
62000	5	27
62237	7	27
62237	13	27
62249	22	27
62249	11	27
62374	3	27
62374	16	27
62485	2	27
62485	17	27
62497	21	27
62497	1	27
62734	6	27
62734	18	27
62746	18	27
62746	6	27
62982	1	27
62982	21	27
62994	17	27

62994	2	27
63231	11	27
63231	22	27
63243	13	27
63243	7	27
63479	5	27
63479	24	27
64000	24	27
64000	5	27
64237	7	27
64237	13	27
64249	22	27
64249	11	27
64388	2	27
64388	17	27
64485	2	27
64485	17	27
64497	21	27
64497	1	27
64734	6	27
64734	18	27
64746	18	27
64746	6	27
64982	1	27
64982	21	27
64994	17	27
64994	2	27
65231	11	27
65231	22	27
65243	13	27
65243	7	27
65479	5	27
65479	24	27
66000	24	27
66000	5	27
66237	7	27
66237	13	27
66249	22	27
66249	11	27
66485	2	27
66485	17	27
66497	21	27
66497	1	27
66734	6	27
66734	18	27
66746	18	27
66746	6	27
66982	1	27
66982	21	27
66994	17	27
66994	2	27
67231	11	27
67231	22	27
67243	13	27
67243	7	27

67479	5	27
67479	24	27
68000	24	27
68000	5	27
68237	7	27
68237	13	27
68249	22	27
68249	11	27
68485	2	27
68485	17	27
68497	21	27
68497	1	27
68734	6	27
68734	18	27
68746	18	27
68746	6	27
68982	1	27
68982	21	27
68994	17	27
68994	2	27
69231	11	27
69231	22	27
69243	13	27
69243	7	27
69479	5	27
69479	24	27
69568	4	27
69568	20	27
70237	7	27
70237	13	27
70485	2	27
70485	17	27
70734	6	27
70734	18	27
70982	1	27
70982	21	27
71231	11	27
71231	22	27
71479	5	27
71479	24	27
72000	19	27
72000	14	27
72249	23	27
72249	10	27
72497	15	27
72497	9	27
72746	20	27
72746	4	27
72994	12	27
72994	8	27
73243	16	27
73243	3	27
74000	19	27
74000	14	27
74237	3	27

74237	16	27
74249	23	27
74249	10	27
74485	8	27
74485	12	27
74497	15	27
74497	9	27
74734	4	27
74734	20	27
74746	20	27
74746	4	27
74982	9	27
74982	15	27
74994	12	27
74994	8	27
75231	10	27
75231	23	27
75243	16	27
75243	3	27
75479	14	27
75479	19	27
76000	19	27
76000	14	27
76237	3	27
76237	16	27
76249	23	27
76249	10	27
76485	8	27
76485	12	27
76497	15	27
76497	9	27
76734	4	27
76734	20	27
76746	20	27
76746	4	27
76763	10	27
76763	23	27
76982	9	27
76982	15	27
76994	12	27
76994	8	27
77231	10	27
77231	23	27
77243	16	27
77243	3	27
77479	14	27
77479	19	27
78000	19	27
78000	14	27
78237	3	27
78237	16	27
78249	23	27
78249	10	27
78485	8	27
78485	12	27

78497	15	27
78497	9	27
78734	4	27
78734	20	27
78746	20	27
78746	4	27
78982	9	27
78982	15	27
78994	12	27
78994	8	27
79231	10	27
79231	23	27
79243	16	27
79243	3	27
79479	14	27
79479	19	27
80000	19	27
80000	14	27
80237	3	27
80237	16	27
80249	23	27
80249	10	27
80485	8	27
80485	12	27
80497	15	27
80497	9	27
80734	4	27
80734	20	27
80746	20	27
80746	4	27
80982	9	27
80982	15	27
80994	12	27
80994	8	27
81231	10	27
81231	23	27
81243	16	27
81243	3	27
81479	14	27
81479	19	27
82237	3	27
82237	16	27
82374	13	27
82374	7	27
82485	8	27
82485	12	27
82734	4	27
82734	20	27
82982	9	27
82982	15	27
83231	10	27
83231	23	27
83479	14	27
83479	19	27
84000	24	27

84000	5	27
84249	22	27
84249	11	27
84497	21	27
84497	1	27
84746	18	27
84746	6	27
84994	17	27
84994	2	27
85243	13	27
85243	7	27
86000	24	27
86000	5	27
86237	7	27
86237	13	27
86249	22	27
86249	11	27
86485	2	27
86485	17	27
86497	21	27
86497	1	27
86734	6	27
86734	18	27
86746	18	27
86746	6	27
86982	1	27
86982	21	27
86994	17	27
86994	2	27
87231	11	27
87231	22	27
87243	13	27
87243	7	27
87479	5	27
87479	24	27
88000	24	27
88000	5	27
88237	7	27
88237	13	27
88249	22	27
88249	11	27
88485	2	27
88485	17	27
88497	21	27
88497	1	27
88734	6	27
88734	18	27
88746	18	27
88746	6	27
88982	1	27
88982	21	27
88994	17	27
88994	2	27
89231	11	27
89231	22	27

```
89243   13  27
89243   7   27
89479   5   27
89479   24  27
89568   18  27
89568   6   27
90000   24  27
90000   5   27
90237   7   27
90237   13  27
90249   22  27
90249   11  27
90485   2   27
90485   17  27
90497   21  27
90497   1   27
90734   6   27
90734   18  27
90746   18  27
90746   6   27
90982   1   27
90982   21  27
90994   17  27
90994   2   27
91231   11  27
91231   22  27
91243   13  27
91243   7   27
91479   5   27
91479   24  27
92000   24  27
92000   5   27
92237   7   27
92237   13  27
92249   22  27
92249   11  27
92485   2   27
92485   17  27
92497   21  27
92497   1   27
92734   6   27
92734   18  27
92746   18  27
92746   6   27
92982   1   27
92982   21  27
92994   17  27
92994   2   27
93231   11  27
93231   22  27
93243   13  27
93243   7   27
93479   5   27
93479   24  27
94237   7   27
```

```
94237   13  27
94485   2   27
94485   17  27
94734   6   27
94734   18  27
94982   1   27
94982   21  27
95231   11  27
95231   22  27
95479   5   27
95479   24  27
96763   22  27
96763   11  27
];

end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Scenario_44 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Scenario_44()

global Scenario_4;

%time   originating node        destination node
Scenario_4 = [
0       19  27
172     23  27
344     15  27
516     20  27
688     12  27
860     16  27
1549    3   27
1721    8   27
1893    4   27
2065    9   27
2237    10  27
2409    14  27
0       14  27
172     10  27
344     9   27
516     4   27
688     8   27
860     3   27
1549    16  27
1721    12  27
1893    20  27
2065    15  27
2237    23  27
2409    19  27
2000    19  27
```

```
2172    23  27
2344    15  27
2516    20  27
2688    12  27
2860    16  27
3549    3   27
3721    8   27
3893    4   27
4065    9   27
4237    10  27
4409    14  27
2000    14  27
2172    10  27
2344    9   27
2516    4   27
2688    8   27
2860    3   27
3549    16  27
3721    12  27
3893    20  27
4065    15  27
4237    23  27
4409    19  27
4000    19  27
4172    23  27
4344    15  27
4516    20  27
4688    12  27
4860    16  27
5549    3   27
5721    8   27
5893    4   27
6065    9   27
6237    10  27
6409    14  27
4000    14  27
4172    10  27
4344    9   27
4516    4   27
4688    8   27
4860    3   27
5549    16  27
5721    12  27
5893    20  27
6065    15  27
6237    23  27
6409    19  27
6000    19  27
6172    23  27
6344    15  27
6516    20  27
6688    12  27
6860    16  27
7549    3   27
7721    8   27
```

```
7893    4   27
8065    9   27
8237    10  27
8409    14  27
6000    14  27
6172    10  27
6344    9   27
6516    4   27
6688    8   27
6860    3   27
7549    16  27
7721    12  27
7893    20  27
8065    15  27
8237    23  27
8409    19  27
8000    19  27
8172    23  27
8344    15  27
8516    20  27
8688    12  27
8860    16  27
9549    3   27
9721    8   27
9893    4   27
10065   9   27
10237   10  27
10409   14  27
8000    14  27
8172    10  27
8344    9   27
8516    4   27
8688    8   27
8860    3   27
9549    16  27
9721    12  27
9893    20  27
10065   15  27
10237   23  27
10409   19  27
12000   24  27
12172   22  27
12344   21  27
12516   18  27
12688   17  27
12860   13  27
13549   7   27
13721   2   27
13893   6   27
14065   1   27
14237   11  27
14409   5   27
12000   5   27
12172   11  27
12344   1   27
```

12516	6	27
12688	2	27
12860	7	27
13549	13	27
13721	17	27
13893	18	27
14065	21	27
14237	22	27
14409	24	27
14000	24	27
14172	22	27
14344	21	27
14516	18	27
14688	17	27
14860	13	27
15549	7	27
15721	2	27
15893	6	27
16065	1	27
16237	11	27
16409	5	27
14000	5	27
14172	11	27
14344	1	27
14516	6	27
14688	2	27
14860	7	27
15549	13	27
15721	17	27
15893	18	27
16065	21	27
16237	22	27
16409	24	27
16000	24	27
16172	22	27
16344	21	27
16516	18	27
16688	17	27
16860	13	27
17549	7	27
17721	2	27
17893	6	27
18065	1	27
18237	11	27
18409	5	27
16000	5	27
16172	11	27
16344	1	27
16516	6	27
16688	2	27
16860	7	27
17549	13	27
17721	17	27
17893	18	27
18065	21	27

18237	22	27
18409	24	27
18000	24	27
18172	22	27
18344	21	27
18516	18	27
18688	17	27
18860	13	27
19549	7	27
19721	2	27
19893	6	27
20065	1	27
20237	11	27
20409	5	27
18000	5	27
18172	11	27
18344	1	27
18516	6	27
18688	2	27
18860	7	27
19549	13	27
19721	17	27
19893	18	27
20065	21	27
20237	22	27
20409	24	27
20000	24	27
20172	22	27
20344	21	27
20516	18	27
20688	17	27
20860	13	27
21549	7	27
21721	2	27
21893	6	27
22065	1	27
22237	11	27
22409	5	27
20000	5	27
20172	11	27
20344	1	27
20516	6	27
20688	2	27
20860	7	27
21549	13	27
21721	17	27
21893	18	27
22065	21	27
22237	22	27
22409	24	27
24000	19	27
24172	23	27
24344	15	27
24516	20	27
24688	12	27

24860	16	27
25549	3	27
25721	8	27
25893	4	27
26065	9	27
26237	10	27
26409	14	27
24000	14	27
24172	10	27
24344	9	27
24516	4	27
24688	8	27
24860	3	27
25549	16	27
25721	12	27
25893	20	27
26065	15	27
26237	23	27
26409	19	27
26000	19	27
26172	23	27
26344	15	27
26516	20	27
26688	12	27
26860	16	27
27549	3	27
27721	8	27
27893	4	27
28065	9	27
28237	10	27
28409	14	27
26000	14	27
26172	10	27
26344	9	27
26516	4	27
26688	8	27
26860	3	27
27549	16	27
27721	12	27
27893	20	27
28065	15	27
28237	23	27
28409	19	27
28000	19	27
28172	23	27
28344	15	27
28516	20	27
28688	12	27
28860	16	27
29549	3	27
29721	8	27
29893	4	27
30065	9	27
30237	10	27
30409	14	27

28000	14	27
28172	10	27
28344	9	27
28516	4	27
28688	8	27
28860	3	27
29549	16	27
29721	12	27
29893	20	27
30065	15	27
30237	23	27
30409	19	27
30000	19	27
30172	23	27
30344	15	27
30516	20	27
30688	12	27
30860	16	27
31549	3	27
31721	8	27
31893	4	27
32065	9	27
32237	10	27
32409	14	27
30000	14	27
30172	10	27
30344	9	27
30516	4	27
30688	8	27
30860	3	27
31549	16	27
31721	12	27
31893	20	27
32065	15	27
32237	23	27
32409	19	27
32000	19	27
32172	23	27
32344	15	27
32516	20	27
32688	12	27
32860	16	27
33549	3	27
33721	8	27
33893	4	27
34065	9	27
34237	10	27
34409	14	27
32000	14	27
32172	10	27
32344	9	27
32516	4	27
32688	8	27
32860	3	27
33549	16	27

33721	12	27
33893	20	27
34065	15	27
34237	23	27
34409	19	27
36000	24	27
36172	22	27
36344	21	27
36516	18	27
36688	17	27
36860	13	27
37549	7	27
37721	2	27
37893	6	27
38065	1	27
38237	11	27
38409	5	27
36000	5	27
36172	11	27
36344	1	27
36516	6	27
36688	2	27
36860	7	27
37549	13	27
37721	17	27
37893	18	27
38065	21	27
38237	22	27
38409	24	27
38000	24	27
38172	22	27
38344	21	27
38516	18	27
38688	17	27
38860	13	27
39549	7	27
39721	2	27
39893	6	27
40065	1	27
40237	11	27
40409	5	27
38000	5	27
38172	11	27
38344	1	27
38516	6	27
38688	2	27
38860	7	27
39549	13	27
39721	17	27
39893	18	27
40065	21	27
40237	22	27
40409	24	27
40000	24	27
40172	22	27

40344	21	27
40516	18	27
40688	17	27
40860	13	27
41549	7	27
41721	2	27
41893	6	27
42065	1	27
42237	11	27
42409	5	27
40000	5	27
40172	11	27
40344	1	27
40516	6	27
40688	2	27
40860	7	27
41549	13	27
41721	17	27
41893	18	27
42065	21	27
42237	22	27
42409	24	27
42000	24	27
42172	22	27
42344	21	27
42516	18	27
42688	17	27
42860	13	27
43549	7	27
43721	2	27
43893	6	27
44065	1	27
44237	11	27
44409	5	27
42000	5	27
42172	11	27
42344	1	27
42516	6	27
42688	2	27
42860	7	27
43549	13	27
43721	17	27
43893	18	27
44065	21	27
44237	22	27
44409	24	27
44000	24	27
44172	22	27
44344	21	27
44516	18	27
44688	17	27
44860	13	27
45549	7	27
45721	2	27
45893	6	27

46065	1	27
46237	11	27
46409	5	27
44000	5	27
44172	11	27
44344	1	27
44516	6	27
44688	2	27
44860	7	27
45549	13	27
45721	17	27
45893	18	27
46065	21	27
46237	22	27
46409	24	27
48000	19	27
48172	23	27
48344	15	27
48516	20	27
48688	12	27
48860	16	27
49549	3	27
49721	8	27
49893	4	27
50065	9	27
50237	10	27
50409	14	27
48000	14	27
48172	10	27
48344	9	27
48516	4	27
48688	8	27
48860	3	27
49549	16	27
49721	12	27
49893	20	27
50065	15	27
50237	23	27
50409	19	27
50000	19	27
50172	23	27
50344	15	27
50516	20	27
50688	12	27
50860	16	27
51549	3	27
51721	8	27
51893	4	27
52065	9	27
52237	10	27
52409	14	27
50000	14	27
50172	10	27
50344	9	27
50516	4	27

50688	8	27
50860	3	27
51549	16	27
51721	12	27
51893	20	27
52065	15	27
52237	23	27
52409	19	27
52000	19	27
52172	23	27
52344	15	27
52516	20	27
52688	12	27
52860	16	27
53549	3	27
53721	8	27
53893	4	27
54065	9	27
54237	10	27
54409	14	27
52000	14	27
52172	10	27
52344	9	27
52516	4	27
52688	8	27
52860	3	27
53549	16	27
53721	12	27
53893	20	27
54065	15	27
54237	23	27
54409	19	27
54000	19	27
54172	23	27
54344	15	27
54516	20	27
54688	12	27
54860	16	27
55549	3	27
55721	8	27
55893	4	27
56065	9	27
56237	10	27
56409	14	27
54000	14	27
54172	10	27
54344	9	27
54516	4	27
54688	8	27
54860	3	27
55549	16	27
55721	12	27
55893	20	27
56065	15	27
56237	23	27

56409	19	27
56000	19	27
56172	23	27
56344	15	27
56516	20	27
56688	12	27
56860	16	27
57549	3	27
57721	8	27
57893	4	27
58065	9	27
58237	10	27
58409	14	27
56000	14	27
56172	10	27
56344	9	27
56516	4	27
56688	8	27
56860	3	27
57549	16	27
57721	12	27
57893	20	27
58065	15	27
58237	23	27
58409	19	27
60000	24	27
60172	22	27
60344	21	27
60516	18	27
60688	17	27
60860	13	27
61549	7	27
61721	2	27
61893	6	27
62065	1	27
62237	11	27
62409	5	27
60000	5	27
60172	11	27
60344	1	27
60516	6	27
60688	2	27
60860	7	27
61549	13	27
61721	17	27
61893	18	27
62065	21	27
62237	22	27
62409	24	27
62000	24	27
62172	22	27
62344	21	27
62516	18	27
62688	17	27
62860	13	27

63549	7	27
63721	2	27
63893	6	27
64065	1	27
64237	11	27
64409	5	27
62000	5	27
62172	11	27
62344	1	27
62516	6	27
62688	2	27
62860	7	27
63549	13	27
63721	17	27
63893	18	27
64065	21	27
64237	22	27
64409	24	27
64000	24	27
64172	22	27
64344	21	27
64516	18	27
64688	17	27
64860	13	27
65549	7	27
65721	2	27
65893	6	27
66065	1	27
66237	11	27
66409	5	27
64000	5	27
64172	11	27
64344	1	27
64516	6	27
64688	2	27
64860	7	27
65549	13	27
65721	17	27
65893	18	27
66065	21	27
66237	22	27
66409	24	27
66000	24	27
66172	22	27
66344	21	27
66516	18	27
66688	17	27
66860	13	27
67549	7	27
67721	2	27
67893	6	27
68065	1	27
68237	11	27
68409	5	27
66000	5	27

66172	11	27
66344	1	27
66516	6	27
66688	2	27
66860	7	27
67549	13	27
67721	17	27
67893	18	27
68065	21	27
68237	22	27
68409	24	27
68000	24	27
68172	22	27
68344	21	27
68516	18	27
68688	17	27
68860	13	27
69549	7	27
69721	2	27
69893	6	27
70065	1	27
70237	11	27
70409	5	27
68000	5	27
68172	11	27
68344	1	27
68516	6	27
68688	2	27
68860	7	27
69549	13	27
69721	17	27
69893	18	27
70065	21	27
70237	22	27
70409	24	27
72000	19	27
72172	23	27
72344	15	27
72516	20	27
72688	12	27
72860	16	27
73549	3	27
73721	8	27
73893	4	27
74065	9	27
74237	10	27
74409	14	27
72000	14	27
72172	10	27
72344	9	27
72516	4	27
72688	8	27
72860	3	27
73549	16	27
73721	12	27

73893	20	27
74065	15	27
74237	23	27
74409	19	27
74000	19	27
74172	23	27
74344	15	27
74516	20	27
74688	12	27
74860	16	27
75549	3	27
75721	8	27
75893	4	27
76065	9	27
76237	10	27
76409	14	27
74000	14	27
74172	10	27
74344	9	27
74516	4	27
74688	8	27
74860	3	27
75549	16	27
75721	12	27
75893	20	27
76065	15	27
76237	23	27
76409	19	27
76000	19	27
76172	23	27
76344	15	27
76516	20	27
76688	12	27
76860	16	27
77549	3	27
77721	8	27
77893	4	27
78065	9	27
78237	10	27
78409	14	27
76000	14	27
76172	10	27
76344	9	27
76516	4	27
76688	8	27
76860	3	27
77549	16	27
77721	12	27
77893	20	27
78065	15	27
78237	23	27
78409	19	27
78000	19	27
78172	23	27
78344	15	27

78516	20	27
78688	12	27
78860	16	27
79549	3	27
79721	8	27
79893	4	27
80065	9	27
80237	10	27
80409	14	27
78000	14	27
78172	10	27
78344	9	27
78516	4	27
78688	8	27
78860	3	27
79549	16	27
79721	12	27
79893	20	27
80065	15	27
80237	23	27
80409	19	27
80000	19	27
80172	23	27
80344	15	27
80516	20	27
80688	12	27
80860	16	27
81549	3	27
81721	8	27
81893	4	27
82065	9	27
82237	10	27
82409	14	27
80000	14	27
80172	10	27
80344	9	27
80516	4	27
80688	8	27
80860	3	27
81549	16	27
81721	12	27
81893	20	27
82065	15	27
82237	23	27
82409	19	27
84000	24	27
84172	22	27
84344	21	27
84516	18	27
84688	17	27
84860	13	27
85549	7	27
85721	2	27
85893	6	27
86065	1	27

86237	11	27
86409	5	27
84000	5	27
84172	11	27
84344	1	27
84516	6	27
84688	2	27
84860	7	27
85549	13	27
85721	17	27
85893	18	27
86065	21	27
86237	22	27
86409	24	27
86000	24	27
86172	22	27
86344	21	27
86516	18	27
86688	17	27
86860	13	27
87549	7	27
87721	2	27
87893	6	27
88065	1	27
88237	11	27
88409	5	27
86000	5	27
86172	11	27
86344	1	27
86516	6	27
86688	2	27
86860	7	27
87549	13	27
87721	17	27
87893	18	27
88065	21	27
88237	22	27
88409	24	27
88000	24	27
88172	22	27
88344	21	27
88516	18	27
88688	17	27
88860	13	27
89549	7	27
89721	2	27
89893	6	27
90065	1	27
90237	11	27
90409	5	27
88000	5	27
88172	11	27
88344	1	27
88516	6	27
88688	2	27

88860	7	27
89549	13	27
89721	17	27
89893	18	27
90065	21	27
90237	22	27
90409	24	27
90000	24	27
90172	22	27
90344	21	27
90516	18	27
90688	17	27
90860	13	27
91549	7	27
91721	2	27
91893	6	27
92065	1	27
92237	11	27
92409	5	27
90000	5	27
90172	11	27
90344	1	27
90516	6	27
90688	2	27
90860	7	27
91549	13	27
91721	17	27
91893	18	27
92065	21	27
92237	22	27
92409	24	27
92000	24	27
92172	22	27
92344	21	27
92516	18	27
92688	17	27
92860	13	27
93549	7	27
93721	2	27
93893	6	27
94065	1	27
94237	11	27
94409	5	27
92000	5	27
92172	11	27
92344	1	27
92516	6	27
92688	2	27
92860	7	27
93549	13	27
93721	17	27
93893	18	27
94065	21	27
94237	22	27
94409	24	27

```
50000   5   27
57194   1   27
64388   2   27
82374   13  27
89568   18  27
96763   22  27
50000   24  27
57194   21  27
64388   17  27
82374   7   27
89568   6   27
96763   11  27
30000   19  27
37194   15  27
44388   12  27
62374   3   27
69568   4   27
76763   10  27
30000   14  27
37194   9   27
44388   8   27
62374   16  27
69568   20  27
76763   23  27
];


end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create_PCAP_Table %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Create_PCAP_Table()
% Creates the table in which the PCAP information is stored for each
% transmission.

global PCAP;
% Packet#   Timer   Status SrcNode     DestNode    RawPacket
PCAP = zeros(1,1021);

end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Spoof %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Spoof()
% Inserting packet for spoofing
```

```
global Spoofed_Node;
global Rec_Node;
global Spoof_Pack_Num;
global Spoof_Packet;
global Energy_Table;
global Node_Status;
global Timer;
global Open_Packets;
global SpNum;
global Packet;
global fileID;
global Packets;
global Node_Status_Table;
global Timer_Spoof;
global Wait_Time;

[cc,dd] = size(Node_Status_Table);
cc = cc+1;

[b,c] = size(Energy_Table);
    d = 0;
for n= 1:b
% We know the node needs to transmit but needs to see if another node
% is transmitting
if Energy_Table(n,1) == Rec_Node
if Node_Status(Energy_Table(n,2),2) == 3;
                d = 1;
end
end
end
% checks ot see if it is able to spoof the node from the checks
% performed above
if  d == 0 && Spoof_Pack_Num > 0 && Timer_Spoof < Timer
        Timer_Spoof = Timer + 1000; % another spoof can occur in 1000ms
        Wait_Time(Rec_Node,2) = 0;

        Spoof_Pack_Num = Spoof_Pack_Num - 1; %can place a limit on the
number of spoofed frames by the user
        [q,r] = size(Open_Packets);
        q = q+1;
        SpNum = q;
        Open_Packets(q,1:7) = [q (Timer+4) 2 1 Spoofed_Node Rec_Node
0]; %log for the open frames
        Node_Status(Rec_Node,2:3) = [5 (Timer+4)]; % continuous node
status log
        Node_Status_Table(cc,1:4) = [Rec_Node,Timer,2,5]; % current
node status log
        Packet(q,1:1016) = Spoof_Packet(1:1016); %creates spoofed frame
        Packets(q,1:3) = [q Spoofed_Node Rec_Node]; %frame is logged
        fprintf('---At Time %i, Packet %i is a spoofed
packet\n',Timer,q);
        fprintf(fileID,'---At Time %i, Packet %i is a spoofed
packet\n',Timer,q);
```

```
end

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Spoofing %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Spoofing()
% Spoofing attack

global Spoofed_Node;
global Rec_Node;
global Routing_Table;
global Spoof_Pack_Num;
global Spoof_Packet;
global Starter_Packet;
global Node_MAC;
global Node_Address;
global SpNum;
global Timer_Spoof;

Timer_Spoof = 0;
SpNum = 0;
Rec_Node = 0;

Spoof_Pack_Num = 100; %limits simulation to 100 injected frames
% not testing a spoofed frame from BS
if Spoofed_Node < 27
    [a,b] = size(Routing_Table);
for n = 1:a %find the next hop to go to from the spoofed node
if Spoofed_Node == Routing_Table(n,1) && Routing_Table(n,2) == 1
            Rec_Node = Routing_Table(n,3);
end
end
% create spoofed frame
    Spoof_Packet = Starter_Packet(1,1:1016);

% Adds Source MAC Address
        Spoof_Packet(17:80) = Node_MAC(Spoofed_Node,2:65);

%Destination MAC Address
        Spoof_Packet(81:144) = Node_MAC(Rec_Node,2:65);

% Source Address
        Spoof_Packet(169:184) =  Node_Address(Spoofed_Node,50:65);

% Destination Address
        Spoof_Packet(185:200) =  Node_Address(27,50:65);

%Sequence Number
```

```
        Spoof_Packet(225:256) = randi([0,1],1,32);

% Padding
        Spoof_Packet(425:864) = randi([0 1],1,440);

% Next Header
        Spoof_Packet(993:1000) = randi([0 1],1,8);

% Payload
        Spoof_Packet(297:424) = randi([0,1],1,128);

% Message Integrity Code
        Spoof_Packet(865:992) = randi([0 1],1,128);

% Field Check Sum (CRC)
%Create the CRC for the Packet
            crc1 = Spoof_Packet(1:1000);
            crc = crc16(crc1);
        Spoof_Packet(1001:1016) = crc;
end

end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DDOS %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function DDOS(n)
% Directed Denial of Service Attack

global DDOS_Node;
global Node_Status;
global Energy_Table;
global Cycles;
global Enc;

% creates node within table to simulate the rogue node to conduct DOS
% attack, initial set-up
if n == 1
    [a,b] = size(Energy_Table);
    Energy_Table((a+1),1:2) = [28,DDOS_Node];
    Enc(DDOS_Node) = 0;

%ensures node remains in transmitting status each cycle and that the
%affected node remains in a receiving status
elseif n == 2
    Node_Status(DDOS_Node,2:3) = [5 Cycles];
    Node_Status(28,1:5) = [28 3 Cycles 0 0];

end
```

```
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MITM %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function MITM(PN)
% Man in the Middle attack

global Packet;
global fileID;
global MITMAtt;

%Alerts user the attack was performed
fprintf('---A MITM attack was executed for packet %3i\n',PN);
fprintf(fileID,'---A MITM attack was executed for packet %3i\n',PN);
%injects random information into the payload
Packet(PN,297:864) = randi([0,1],1,568);
crc = crc16(Packet(PN,1:1000));
%adjusts the crc to properly validate at follow-on node
Packet(PN,1001:1016) = crc(1:16);

[aa,bb] = size(MITMAtt);
cc = 0;
%Logs the frames in which the MITM was conducted on to analyze
afterwards
for n = 1:bb
if MITMAtt(n) == PN
      cc = 1;
end
end
if cc == 0
    MITMAtt(n + 1) = PN;
end

end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Run Simulation %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Run_Simulation()
% This function iterates every 1 ms to simulate a network environment.

global Timer;
global Events;
global DDOS_Node;
global Cycles;
global fileID;
global Spoofed_Node;
global Rec_Node;
global Node_Status;

%Gets last time a sensing event is obtained
LastEvent = max(Events);

%Adds time for the sesning event to be completed
Cycles = LastEvent(1) + 10000;   %adds 10 seconds

for n = 1:Cycles
%Checks for errors from preious cycle and distributes the errors as
%required
    Check_For_Errors(Timer);

%Adjusts the Node's status once it has reached the end of its process
    Check_Node_Status(Timer);

    a = randi(100,1);
if Spoofed_Node < 27
if a <= 1 && Node_Status(Rec_Node,2) == 2 &&
Node_Status(Spoofed_Node,2) == 0
              Spoof();
end
end

%Cycles through open packets to see if it is ready for the cycle
    Check_For_Open_Packets(Timer);

%Checks for new sensing events to be added to an open packet
    Check_For_New_Events(Timer);

%Adds the time to the specific energy use tables
xxx = mod(Timer,5000);

%alerts user the program is running and how far along it is
xxxx = Timer /1000;
if xxx == 0
    fprintf('%i seconds have elapsed \n',xxxx);
    fprintf(fileID,'%i seconds have elapsed \n',xxxx);
end

% checks status of DOS rogue and affected node
if DDOS_Node < 28
        DDOS(2);
end

%Logs energy use of the nodes at the time in the cycle
      Check_For_Energy_Use();
    Timer = Timer + 1;
    LastEvent = max(Events);

%ensures last event is complete before ending the program
```

```
    Cycles = LastEvent(1) + 5000;
end

end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check_For_Errors %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Check_For_Errors(Timer)
% Checks all of the Open_Packets to show that there was an error in
% receiving a packet. It is documented that the packet was received in
% error but the node will continue its same processing sequence, it
% will just fail in processing and the transmitting node will have to
% transmit again.

%{
This is for me while programming to keep from checking another file....
Packet #   timer   event   # of trans     TransNode RecNode     Errors
Open_Packets = [
    0       0       0           0           0           0        0
    ];
%}

global Open_Packets;
global Energy_Table;

[a,b] = size(Open_Packets);   %gets the number of the amount of open
packets
for n = 1:a     % cycles through all of the open packets
    c = 0;      % variable
if Open_Packets(n,3) == 2   % Checks if a packet is being transmitted
for m = 1:a     % If a packet is being transmitted it is then checked
against all other open packets
if Open_Packets(m,3) == 2 && Open_Packets(n,5) ~= Open_Packets(m,5)  %
If will cycle itself again to see if another node is transmitting
% If another node is transmitting then it needs to check if
% the receiving node is affected by cycling through all of
% the potentialling affected nodes.
            [d,e] = size(Energy_Table);   % Gets the size of the table
for p = 1:d                     % Cycles through the table
if Open_Packets(m,5) == Energy_Table(p,1)
                        % Finds in the table the affected nodes
if Energy_Table(p,2) == Open_Packets(n,6)
                        % If a receiving node matches an affected node
                          diff=(Open_Packets(m,2) - Open_Packets(n,2));
                          check = power(diff,2);
if check < 17
                            Open_Packets(n,3) = 3;
                        % the error is placed in the original packet
end
end
```

```
end
if Open_Packets(n,5) == Energy_Table(p,1)
                        % Finds in the table the affected nodes
if Energy_Table(p,2) == Open_Packets(m,6)
                        % If a receiving node matches an affected node
                          diff=(Open_Packets(m,2) - Open_Packets(n,2));
                          check = power(diff,2);
if check < 17
                             Open_Packets(m,3) = 3;
                        % the error is placed in the original packet
end
end
end
end
end
end
end
end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check_Node_Status %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Check_Node_Status(Timer)
%Adjusts the node status to the next step when the node timer is
reached

global Node_Status;
global Wait_Time;
global Open_Packets;
global Node_Status_Table;

for n=1:27      %Cycle through all of the nodes
if Wait_Time(n,2) == 25;
            % If wait time reaches 25 ms then it will go to sleep
            Wait_Time(n,2) = 0;
            % Resets wait time for next evolution
            Node_Status(n,2) = 6;   % Status 6 is going to sleep
                [A,B] = size(Node_Status_Table);
                Node_Status_Table((A+1),1:4) = [n Timer 2 6];
            Node_Status(n,3) = Timer + 1;
            % It takes 1 ms to go to sleep

end

% if the node is at 2 and still waiting, add 1 ms to energy use
if Node_Status(n,2) == 2  && Node_Status(n,1) ~= 27

            Wait_Time(n,2) = Wait_Time(n,2) + 1;
            % Adds 1 ms to the Wait time
elseif Node_Status(n,3) <= Timer            % Checks if the node is
ready to transition to the next stage
```

```
% If node is either at:
%  0 = Node is asleep
%  1 = finished waking up
%  3 = finished transmitting
%  4 = finished Processing
%  5 = Finished receiving
%  6 = Going to sleep
%  7 = Waiting for Post Trans
if Node_Status(n,2) == 1
                    Node_Status(n,2) = 2;
            % change node to a waiting status making it available for
            % the next function
                        [A,B] = size(Node_Status_Table);
                        Node_Status_Table((A+1),1:4) = [n Timer 1 2];
                        % entry into node status log
elseif Node_Status(n,2) == 7
                    Node_Status(n,2) = 2;
                  % it is now in waiting phase to begin retransmission
                        [A,B] = size(Node_Status_Table);
                        Node_Status_Table((A+1),1:4) = [n Timer 7 2];
                        % entry into node status log
                        Wait_Time(n,2) = 0;
elseif Node_Status(n,2) == 4
                    Node_Status(n,2) = 2;
                  % it is now in waiting phase to begin retransmission
                        [A,B] = size(Node_Status_Table);
                        Node_Status_Table((A+1),1:4) = [n Timer 4 2];
                        % entry into node status log
                        Wait_Time(n,2) = 0;
elseif Node_Status(n,2) == 5
                  % node is finished receiving
                    Node_Status(n,2) = 4;
                  % node is now processing
                        [A,B] = size(Node_Status_Table);
                        Node_Status_Table((A+1),1:4) = [n Timer 5 4];
                        % entry into node status log
                        Wait_Time(n,2) = 0;
elseif Node_Status(n,2) == 3
                  % if node is finished transmitting
                    Node_Status(n,2) = 7;
                  % it is now in a processing phase
                        [A,B] = size(Node_Status_Table);
                        Node_Status_Table((A+1),1:4) = [n Timer 3 7];
                        % entry into node status log
                        Wait_Time(n,2) = 0;
                    [c,d] = size(Open_Packets);
for aa = 1:c
if Open_Packets(aa,5) == Node_Status(n,1)
if Open_Packets(aa,4)==1 || Open_Packets(aa,4)==5
                                Node_Status(n,3) = Timer + 6;
                              % Time after first transmission
% seed number inserted for BS neighbor
% nodes
if Node_Status(n,1) == 25 || Node_Status(n,1) == 26
```

```
                                     Node_Status(n,3) =
Node_Status(n,3) + randi([1,4],1,1);
end

elseif Open_Packets(aa,4) == 2 || Open_Packets(aa,4) == 6
                                Node_Status(n,3) = Timer + 10; % Time
after second transmission
if Node_Status(n,1) == 25 || Node_Status(n,1) == 26
                                     Node_Status(n,3) =
Node_Status(n,3) + randi([1,4],1,1); %seed number
end

elseif Open_Packets(aa,4) == 3 || Open_Packets(aa,4) == 7
                                Node_Status(n,3) = Timer + 20; % Time
after third transmission
if Node_Status(n,1) == 25 || Node_Status(n,1) == 26
                                     Node_Status(n,3) =
Node_Status(n,3) + randi([1,4],1,1);%seed number
end
elseif Open_Packets(aa,4) == 4 || Open_Packets(aa,4) == 8
                                Node_Status(n,3) = Timer + 40; % Time
after fourth transmission
if Node_Status(n,1) == 25 || Node_Status(n,1) == 26
                                     Node_Status(n,3) =
Node_Status(n,3) + randi([1,4],1,1);%seed number
end
end
end
end

end
if Node_Status(n,2) == 6
                  %if node has completed its going to sleep phase
                    Node_Status(n,2) = 0;   % Node is now in sleep
                        [A,B] = size(Node_Status_Table);
                        Node_Status_Table((A+1),1:4) = [n Timer 6 0];
                        % entry into node status log
end
end

end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check_For_Open_Packets %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Check_For_Open_Packets(Timer)

% Check to see if a packet is ready to move onto the next function
%Packet #   timer   event   # of trans     TransNode RecNode     Errors
%Open_Packets = [
%    0       0       0           0           0           0        0
%    ];

%   Node#   Status   Stat Time chg  Errors?
%Node_Status = [
%    1       0           0               0;

global Open_Packets;
global Node_Status;
global Node_Status_Table;

[a,b] = size(Open_Packets);   %gets the number of the amount of open
packets
for n = 1:a     % cycles through all of the open packets

if Open_Packets(n,2) <= Timer  && Open_Packets(n,7) <= 50
% Checks if a packet is ready for next function


%Packet is ready to transition from Wakeup
if Open_Packets(n,3) == 0 && Open_Packets(n,1) > 0
% Packet is made but has not been processed

% if packet was in the wake up stage, it needs to move to processing
% the packet
%In order to move to processing the packet, the node must be waiting,
%therefore must check if node is in phase 2, waiting.

%Node is ready for the open packet to process
if Node_Status(Open_Packets(n,5), 2) == 2 &&
Node_Status(Open_Packets(n,5), 3) <= Timer
                   Node_Status(Open_Packets(n,5), 2) = 4;
% Node is now processing original packet
                        [A,B] = size(Node_Status_Table);
                        Node_Status_Table((A+1),1:4) =
[Open_Packets(n,5) Timer 2 4];
                   Node_Status(Open_Packets(n,5),3) = Timer + 8;
% Time to process for Node
                   Open_Packets(n,2) = Timer + 8;
% Time to process the packet
                   Open_Packets(n,3) = [2];
% Adjusts packet status that it is now in the processing phase

end

% Packet has finished processing and ready for transmission
elseif Open_Packets(n,3) == 1 && Open_Packets(n,2) <= Timer
% Node can only go to next function if it is in state 2,
% otherwise no need to check other functions
if Node_Status(Open_Packets(n,5), 2) == 4 &&
Node_Status(Open_Packets(n,5), 3) <= Timer

% Function to transmit the packet and determine if it
% can transmit, this will contain a lot of code to
% determine the energy of the affected nodes and etc.

%Will need to place nodes in either receiving or
%transmitting phases and then all into a processing
%phase.
                   Transmission(Open_Packets(n,1));

elseif Node_Status(Open_Packets(n,5), 2) == 2 &&
Node_Status(Open_Packets(n,5), 3) <= Timer
                   Transmission(Open_Packets(n,1));
end

% Packet has completed transmission and is ready to be post
% processed by the Receiving Node
elseif Open_Packets(n,3) == 2 && Open_Packets(n,2) <= Timer
%Check Packet Status
                Open_Packets(n,3) = 3;
                Process_Received_Packet(Open_Packets(n,1));
% Sends to function to process the packet

% Packet has completed processing and needs to be
% transmitted
elseif Open_Packets(n,3) == 3 && Open_Packets(n,2) <= Timer
% Checks node status to see if it is ready to transmit
if Node_Status(Open_Packets(n,5),2 ) == 2 &&
Node_Status(Open_Packets(n,5), 3) <= Timer
                    Transmission(Open_Packets(n,1));
end

end

end

end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Process_Received_Packet %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Process_Received_Packet(Packet_Number)
%Adjusts the node status to the next step when the node timer is
% reached

global Node_Status;
global Packet;
global Open_Packets;
global Timer;
global Node_Address;
global Node_MAC;
global Energy_Table;
global Node_Status_Table;
global MITM_Nodes;
global fileID;
global MITMAtt;
global SpNum;
global Rec_Node;

PN = Packet_Number;
source = 0;
aaa = 0;

% Checks for Receiving nodes status to see if it able to begin
% processingand if its timer is good

if Node_Status(Open_Packets(PN,6), 2) == 4 &&
Node_Status(Open_Packets(PN,6), 3) <= Timer
% Check to see if there was an error for the packet or node
if Open_Packets(PN,7) <= 49 && Node_Status(Open_Packets(PN,6),4) <= 99
% No errors were found, receiving node is ready for processing
               Node_Status(Open_Packets(PN,6),3) = Timer + 5;
% Sets timer on the node
               Open_Packets(PN,2) = Timer + 5;
% Time to process the packet
               Open_Packets(PN,3) = 3;
% Sets status of the Packet to have been processed

if MITM_Nodes(1) == Open_Packets(PN,5) && MITM_Nodes(2) ==
Open_Packets(PN,6)
                    MITMAtt(1) = 0;
                    MITM(PN);
end
%               % Add BER
                Test_Packet(1,1:1016) = Packet(PN,1:1016);
if SpNum == PN && Rec_Node == Open_Packets(PN,6)
%          % Spoof skipped BER in order to properly inject the frame
else
for pp = 1:1016
                        bb = Packet(PN,pp);
                        cc = randi(100000,1);
%performs the random BER on the frame being received
if bb == 0 && cc <= 1
                            fprintf('At time %i a BER was inserted for
Packet %i and bit # %i \n', Timer, PN, pp);
                            fprintf(fileID,'At time %i a BER was
inserted for Packet %i and bit # %i \n', Timer, PN, pp);
                            Test_Packet(pp) = 1;
elseif bb == 1 && cc <=1
                            fprintf('At time %i a BER was inserted for
Packet %i and bit # %i \n', Timer, PN, pp);
                            fprintf(fileID,'At time %i a BER was
inserted for Packet %i and bit # %i \n', Timer, PN, pp);
                            Test_Packet(pp) = 0;
end
end
end
%Check addresses
```

```
                destmac(1:64) = Packet(PN,81:144);
                dm = 0;
                srcmac(1:64) = Packet(PN,17:80);
                sm = 0;
                destip(1:16) = Packet(PN,185:200);
                di = 0;


% checks to make sure addresses transmitted correctly
for ad = 1:27
if srcmac(1:64) == Node_MAC(ad,2:65)
if Open_Packets(PN,5) == ad
                           sm = 1;
end
end
if destmac(1:64) == Node_MAC(ad,2:65)
if Open_Packets(PN,6) == ad
                            dm = 1;
end
end
end
if destip(1:16) == Node_Address(27,50:65)
                    di = 1;
end

                tot = sm + dm + di;

%          % Checks CRC
            crc = crc16(Test_Packet(1:1000));
if Packet(PN,1001:1016) == crc(1:16)
                aaa = 1;
else
                fprintf('At time %i CRC Failed for Packet %i \n',
Timer, PN);
                fprintf(fileID,'At time %i CRC Failed for Packet %i
\n', Timer, PN);
end

% Sends to function to change the packet headers for the
% next transmission if it is not at the Base Station

               source = Open_Packets(PN,5);

%if everything checks out with addresses and CRC
if Open_Packets(PN,6) ~= 27 && aaa == 1 && tot == 3
                   Packet_Next_Hop(Open_Packets(PN,1));
                   crc = crc16(Packet(PN,1:1000));
                   Packet(PN,1001:1016) = crc;
                   Open_Packets(PN,4) = 0;
%if received by the BS and an error has occured logs the
%error or logged as received
elseif Open_Packets(PN,6) == 27 && Open_Packets(PN,4) == 1
if aaa == 1 && tot == 3
                       Open_Packets(PN,7) = 99; % received no error
```

```
else
                       Open_Packets(PN,7) = 96; % received with error
end
end
end

else

end% Node_Status(Open_Packets(n,6), 2) == 2 &&
Node_Status(Open_Packets(n,6), 3) <= Timer

%Now to consider the nodes that were affected

   [b,c] = size(Energy_Table);
%Cycles through the table again to set statuses
for n = 1:b

% Checks for sending node
if Energy_Table(n,1) ==  source;

                AN = Energy_Table(n,2); % Affected Node

% Node will now process the packet and discard since it is
% not to the node
if Node_Status(AN,2) == 4 && Open_Packets(PN,6) ~= AN

                    Node_Status(AN,3) = Timer + 5;
% Adjusts time for nodes next function

% Node is waiting to receive packet from receiving node to confirm that
the packet was successfully forwarded
elseif Node_Status(AN,2) == 7
                    Node_Status(AN,2) = 4;
%Switch Node Status to Receiving
                        [A,B] = size(Node_Status_Table);
                        Node_Status_Table((A+1),1:4) = [AN Timer 7 4];
                    Node_Status(AN,3) = Timer + 5;
% Adjusts time for node's next function

end% if Node_Status(Open_Packets(a,6),2) == 2



end% Energy_Table(n,1) == a
end% for n = 1:b


end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Transmission %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Transmission(Packet_Number)
% Function to transmit the packet and determine if it
% can transmit, this will contain a lot of code to
% determine the energy of the affected nodes and etc.

%Will need to place nodes in either receiving or
%transmitting phases and then all into a processing
%phase.

global Open_Packets;
global Energy_Table;
global Node_Status;
global PCAP;
global Timer;
global Packet;
global Node_Status_Table;
global Routing_Table;
global Node_MAC;
global Node_Address;
global Events;
global Packet_Retrans_Tracker;
global NodeTX;
global NodeRX;

a = Packet_Number;
PN = a;
d = 0;
SMA = 0;
SIP = 0;

if Open_Packets(PN,4) == 4
[f,b] = size(Routing_Table);
% checks for which route it should use
    SMAC = Packet(PN,17:80);
    SIPC = Packet(PN,169:184);
for nn = 1:f
if Open_Packets(PN,5) == Routing_Table(nn,1) && 2 ==
Routing_Table(nn,2)
            Open_Packets(PN,6) = Routing_Table(nn,3);
            Packet(PN,81:144) = Node_MAC(Open_Packets(PN,6),2:65);
            nn = f;
end

end
for mm = 1:27
if SMAC(1:64) == Node_MAC(mm,2:65)
            SMA = mm;
end

if SIPC == Node_Address(mm,50:65)
            SIP = mm;
```

```
end
end
if SIP == SMA
        Packet(PN,162) = [2];
else
        Packet(PN,161) = [2];
end

%Adjust new CRC
     crc = crc16(Packet(PN,1:1000));
     Packet(PN,1001:1016) = crc(1:16);

end

%Frame had a collision on the last hop
if Open_Packets(PN,4) >= 1 && Open_Packets(PN,6) == 27
    Open_Packets(PN,7) = 97;
    d = 1;
end

% frame has been transmitted 8 times and is going into storage for
% futuretransmission
if Open_Packets(PN,4) >= 8 && Open_Packets(PN,7) <= 49
    Open_Packets(PN,7) = 98;
    [j,k] = size(Events);
if j <= 3000
                tim = Timer + 1000 + randi(200,1);
                [aa,bb] = size(Packet_Retrans_Tracker);
                Packet_Retrans_Tracker((aa+1),1:4) = [PN, 00,
Open_Packets(PN,5), Timer];
                Events((j+1),1:3) = [tim Open_Packets(PN,5) 27];
end
    d = 1;
end

[b,c] = size(Energy_Table);

%variable used to determine if another node is transmitting, if it is
%transmitting d will equal 1, if any of the nodes are not transmitting
% then the transmission can occur.

for n= 1:b
% We know the node needs to transmit but needs to see if another
% node is transmitting
if Energy_Table(n,1) == Open_Packets(a,5)
if Node_Status(Energy_Table(n,2),2) == 3;
            d = 1;
if Open_Packets(a,5) == 25 || Open_Packets(a,5) == 26
                 Open_Packets(a,2) = Timer + 10;
else
                Open_Packets(a,2) = Timer + 20;
end
end
end
```

```
end

% If there isn't a surrounding node that is transmitting the node may
% begin  to transmit.
if d == 0
%switch Transmitting node to transmitting status
    Node_Status(Open_Packets(a,5),2) = 3;
    [A,B] = size(Node_Status_Table);
    Node_Status_Table((A+1),1:4) = [Open_Packets(a,5) Timer 2 3];
    [g,h] = size(PCAP);
    k = g + 1;
    PCAP(k, 1:1021) = [a Timer Open_Packets(a,5) Open_Packets(a,6)
Open_Packets(a,4) Packet(a,1:1016)];
    NodeTX(Open_Packets(a,5),2) = NodeTX(Open_Packets(a,5),2) + 1;

%Adjust timers for transmitting nodes
    Node_Status(Open_Packets(a,5),3) = Timer + 7;

    Open_Packets(a,4) = Open_Packets(a,4) + 1;
    Open_Packets(a,3) = 2; %Adjust packet status to transmitting
    Open_Packets(a,2) = Timer + 7; %Adjust packet time

%Cycles through the table again to set statuses
for n = 1:b

% Checks for sending node
if Energy_Table(n,1) ==  Open_Packets(a,5)
%Checks for affected nodes
if Energy_Table(n,2) == Open_Packets(a,6)



%When the receiving node is found, must check its status.
% If receiving node is ready to receive, the status is
% moved to receiving
if Node_Status(Open_Packets(a,6),2) == 2 ||
Node_Status(Open_Packets(a,6),2) == 7
% Receiving node is waiting and ready to receive
                    NodeRX(Open_Packets(a,6),2) =
NodeRX(Open_Packets(a,6),2) + 1;
                    Node_Status(Open_Packets(a,6),2) = 5;
% Switches Node Status to Receiving
                        [A,B] = size(Node_Status_Table);
                         Node_Status_Table((A+1),1:4) =
[Open_Packets(a,6) Timer 2 5];
                    Node_Status(Open_Packets(a,6),3) = Timer + 7;
% Adjusts time for nodes next function



%Wake up the node
elseif Node_Status(Open_Packets(a,6),2) == 0
                    Node_Status(Open_Packets(a,6),2) = 1;
```

```
%Node is now being woken up
                        [A,B] = size(Node_Status_Table);
                        Node_Status_Table((A+1),1:4) =
[Open_Packets(a,6) Timer 0 1];
                    Node_Status(Open_Packets(a,6),3) = Timer + 1;
% Time for node to wake up


%Node is going to sleep
elseif Node_Status(Open_Packets(a,6),2) == 6
                    Node_Status(Open_Packets(a,6),2) = 1;
%Node is now being woken up
                        [A,B] = size(Node_Status_Table);
                        Node_Status_Table((A+1),1:4) =
[Open_Packets(a,6) Timer 6 1];
                    Node_Status(Open_Packets(a,6),3) = Timer + 1; %
Time for node to wake back up

end% if Node_Status(Open_Packets(a,6),2) == 2

end% Energy_Table(n,2) == Open_Packets(a,6)

if Energy_Table(n,2) ~= Open_Packets(a,6)
                AN = Energy_Table(n,2); % Affected Node

%When the receiving node is found, must check its status.
% If receiving node is ready to receive, the status is
% moved to receiving
if Node_Status(AN,2) == 2
                    NodeRX(AN,2) = NodeRX(AN,2) + 1;
                    Node_Status(AN,2) = 5;
%Switches Node Status to Receiving
                        [A,B] = size(Node_Status_Table);
                        Node_Status_Table((A+1),1:4) = [AN Timer 2 5];
                    Node_Status(AN,3) = Timer + 7;
% Adjusts time for nodes next function



%Wake up the node
elseif Node_Status(AN,2) == 0
                    Node_Status(AN,2) = 1;
%Node is now being woken up
                        [A,B] = size(Node_Status_Table);
                        Node_Status_Table((A+1),1:4) = [AN Timer 0 1];
                    Node_Status(AN,3) = Timer + 1;
% Time for node to wake up

%Node is receving another packet
elseif Node_Status(AN,2) == 5


%Node is going to sleep
```

```
elseif Node_Status(Open_Packets(a,6),2) == 6
                    Node_Status(AN,2) = 1;
%Node is now being woken up
                        [A,B] = size(Node_Status_Table);
                        Node_Status_Table((A+1),1:4) = [AN Timer 6 1];
                    Node_Status(AN,3) = Timer + 1;
% Time for node to wake back up

% Node is waiting to receive packet from receiving node to confirm that
the packet was successfully forwarded
elseif Node_Status(AN,2) == 7
                    Node_Status(AN,2) = 5;
%Switches Node Status to Receiving
                    NodeRX(AN,2) = NodeRX(AN,2) + 1;
                        [A,B] = size(Node_Status_Table);
                        Node_Status_Table((A+1),1:4) = [AN Timer 7 5];
                    Node_Status(AN,3) = Timer + 7;
% Adjusts time for nodes next function

end% if Node_Status(Open_Packets(a,6),2) == 2

end% Energy_Table(n,2) ~= Open_Packets(a,6)

end% Energy_Table(n,1) ==  Open_Packets(n,5)
end% for n = 1:b

end% function



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Packet_Next_Hop %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Packet_Next_Hop(Packet_Number)
% Takes the packet and adjusts the source and dest MAC addresses
% Adjusts the Next_Hop field

global Packet;
global Open_Packets;
global Routing_Table;
global Node_MAC;

PN = Packet_Number;

[a,b] = size(Routing_Table);
cat = 1;
ch = 0;
% checks for which route it should use
for n = 1:a
% checks for source in first col
% checks for primary route
% checks that the source is not the destination
```

```
if Open_Packets(PN,6) == Routing_Table(n,1) && cat ==
Routing_Table(n,2) && ch == 0
%do not send to frame to node it previously received it from to
%prevent looping
if Open_Packets(PN,5) == Routing_Table(n,3) && ch == 0
                cat = 2;
else
%changes out source and destination addresses for the next hop
            Open_Packets(PN,5) = Open_Packets(PN,6);
            Open_Packets(PN,6) = Routing_Table(n,3);
            Packet(PN,17:80) = Node_MAC(Open_Packets(PN,5),2:65);
            Packet(PN,81:144) = Node_MAC(Open_Packets(PN,6),2:65);
            Hop_Limit = bi2de(Packet(PN,163:168));
            HLB = Hop_Limit - 1; %reduces hop limit field
            Packet(PN,163:168) = de2bi(HLB,6);
            ch = 1;
            n = a;
end
end
end

end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check_For_New_Events %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Check_For_New_Events(Timer)
% This function will create the new packet and check the status of the
% nodeto determine if it is busy, asleep, or in a waiting phase in
% which it canproce4ss the packet

global Events;
global Open_Packets;
global Packet;
global Starter_Packet;
global Node_MAC;
global Node_Address;
global Node_Status;
global Routing_Table;
global Node_Status_Table;
global Original_Packet;
global Orig_Events;
global Packet_Retrans_Tracker;
global Payload;
global MAC;
global Packets;

% This section Creates the Packet
    [e,f] = size(Events);
```

```
for n = 1:e
% Is the frame one of the original events or was it being stored to
% be retransmitted later?
if Timer == Events(n,1) && n <= Orig_Events % Orinial event

            [a,b] = size(Packet);
%gets the number of the amount of open packets
            [q,r] = size(Open_Packets);
            [ss,tt] = size(Packets);
            ss=ss+1;

            c = a + 1;
            q = q + 1;



%Creates new standard packet
            Packet(c,1:1016) = Starter_Packet(1,1:1016);

%Need to find next hop MAC Address
            Next_Hop(1:64) = ones(1,64);
            Source(1:64) = ones(1,64);
            [z,y] = size(Routing_Table);
            Next_Node = 0;
for p = 1:z
if Events(n,2) == Routing_Table(p,1) && Routing_Table(p,2) == 1
                    Next_Hop = Node_MAC(Routing_Table(p,3),2:65);
                    Next_Node = Routing_Table(p,3);
                    Source = Node_MAC(Events(n,2),2:65);
end
end

% Adds Source MAC Address
            Open_Packets(q,5) = [Events(n,2)];

            Packet(c,17:80) = Source;

%Destination MAC Address
            Open_Packets(q,1:7) = [q Timer 0 0 Events(n,2) Next_Node
0];
            Packets((ss),1) = q;
            Packets((ss),2) = Events(n,2);
            Packets((ss),3) = Next_Node;

            Packet(c,81:144) = Next_Hop;

% Source Address
            Packet(c,169:184) =  Node_Address(Events(n,2),50:65);

% Destination Address
            Packet(c,185:200) =  Node_Address(27,50:65);

%Sequence Number
            Packet(c,225:256) = de2bi([Node_Status(Events(n,2),5)],32);
            Node_Status(Events(n,2),5) = Node_Status(Events(n,2),5) +
1;

% Padding
            Packet(c,425:864) = randi([0 1],1,440);

% Next Header
            Packet(c,993:1000) = randi([0 1],1,8);

% Payload
            Encryption_Decryption(c,1);
            Packet(c,297:424) = Payload(c,1:128);

% Message Integrity Code
            Packet(c,865:992) = MAC(1:128);

% Field Check Sum (CRC)
%Create the CRC for the Packet
                crc1 = Packet(c,1:1000);
                crc = crc16(crc1);
            Packet(c,1001:1016) = crc;

% Create the original frame for comparrison
            [g,h] = size(Packet);
            Original_Packet(g,:) = Packet(c,:);



% Node is awake and ready to send frame once processed
if Node_Status(Events(n,2),2) == 2
                Open_Packets(q,7) = [2];
                Open_Packets(q,2) = Timer + 8;
                Node_Status(Events(n,2),2) = 4;
                Node_Status(Events(n,2),3) = Timer + 8;
                    [A,B] = size(Node_Status_Table);
                    Node_Status_Table((A+1),1:4) = [Open_Packets(q,5)
Timer 2 4];

%node is asleep or going to sleep, needs to be woken up
elseif Node_Status(Events(n,2),2) == 0 || Node_Status(Events(n,2),2) ==
6
                Open_Packets(q,7) = [0];
                Open_Packets(q,2) = Timer + 1;
                Node_Status(Events(n,2),2) = 1;
                Node_Status(Events(n,2),3) = Timer + 1;
                    [A,B] = size(Node_Status_Table);
                    Node_Status_Table((A+1),1:4) = [Open_Packets(q,5)
Timer 0 1];
end

% Frame was stored and now being retransmitted
elseif Timer == Events(n,1) && n > Orig_Events
% log to track retransmitted frames
            [dd,ee] = size(Packet_Retrans_Tracker);
```

```
% transfers the information from the original frame to the
% new retransmitted frame
for ff = 1:dd
if Packet_Retrans_Tracker(ff,3) ==  Events(n,2) &&
Packet_Retrans_Tracker(ff,2) == 0
                        [q,r] = size(Open_Packets);
                        q = q + 1;
                        Packet_Retrans_Tracker(ff,2) = q;
                        [a,b] = size(Packet);
%gets the number of the amount of open packets
                        c = a + 1;
                        [ss,tt] = size(Packets);
                        ss=ss+1;
%Creates new standard packet
                        Packet(c,1:1016) = Starter_Packet(1,1:1016);
%Need to find next hop MAC Address
                        Next_Hop(1:64) = ones(1,64);
                        Source(1:64) = ones(1,64);
                        [z,y] = size(Routing_Table);
                        Next_Node = 0;
for p = 1:z
if Events(n,2) == Routing_Table(p,1) && Routing_Table(p,2) == 1
                                Next_Hop =
Node_MAC(Routing_Table(p,3),2:65);
                                Next_Node = Routing_Table(p,3);
                                Source = Node_MAC(Events(n,2),2:65);
end
end


                        Open_Packets(q,5) = [Events(n,2)];

                        Packet(c,17:80) = Source;

%Destination MAC Address
                        Open_Packets(q,1:7) = [q Timer 0 0 Events(n,2)
Next_Node 0];
                        Packets((ss),1) = q;
                        Packets((ss),2) = Events(n,2);
                        Packets((ss),3) = Next_Node;

                        Packet(c,81:144) = Next_Hop;

% Hop limit and path indication bits
                        Packet(c,161:168) =
Packet(Packet_Retrans_Tracker(ff,1),161:168);

% Source Address
                        Packet(c,169:184) =
Packet(Packet_Retrans_Tracker(ff,1),169:184);

% Destination Address
```

```
                        Packet(c,185:200) =
Packet(Packet_Retrans_Tracker(ff,1),185:200);

%Sequence Number
                        Packet(c,225:256) =
Packet(Packet_Retrans_Tracker(ff,1),225:256);


% Payload
%Packet(c,297:424) = ones(1,128);
                        Packet(c,297:424) =
Packet(Packet_Retrans_Tracker(ff,1),297:424);

% Padding
                        Packet(c,425:864) =
Packet(Packet_Retrans_Tracker(ff,1),425:864);
%             Packet(c,425:864) = zeros(1,440);

% Message Integrity Code
                        Packet(c,865:992) =
Packet(Packet_Retrans_Tracker(ff,1),865:992);
%             Packet(c,865:992) = ones(1,128);

% Next Header
                        Packet(c,993:1000) =
Packet(Packet_Retrans_Tracker(ff,1),993:1000);

% Field Check Sum (CRC)
%Create the CRC for the Packet
                            crc1 = Packet(c,1:1000);
                            crc = crc16(crc1);
                        Packet(c,1001:1016) = crc;

% store as an original packet for comparison
                        [g,h] = size(Packet);
                        Original_Packet(g,:) = Packet(c,:);

% node is awake and ready to transmit
if Node_Status(Events(n,2),2) == 2
                            Open_Packets(q,7) = [2];
                            Open_Packets(q,2) = Timer + 8;
                            Node_Status(Events(n,2),2) = 4;
                            Node_Status(Events(n,2),3) = Timer + 8;
                                [A,B] = size(Node_Status_Table);
                                Node_Status_Table((A+1),1:4) =
[Open_Packets(q,5) Timer 2 4];

%node is either asleep or going to sleep,
%needs to be woken up
elseif Node_Status(Events(n,2),2) == 0 || Node_Status(Events(n,2),2) ==
6
                            Open_Packets(q,7) = [0];
                            Open_Packets(q,2) = Timer + 1;
                            Node_Status(Events(n,2),2) = 1;
                            Node_Status(Events(n,2),3) = Timer + 1;
                                [A,B] = size(Node_Status_Table);
                                Node_Status_Table((A+1),1:4) = 
[Open_Packets(q,5) Timer 0 1];
end
end
end
end
end
end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check_For_Energy_Use %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Check_For_Energy_Use()
%At the end of the evolution the time is added for each node depending 
on
%what stage it is in at that time

global Node_Status;
global Energy_Use;

for n=1:27      %Cycle through all of the nodes

if Node_Status(n,2) == 0    % Node Asleep
            Energy_Use(n,9) = Energy_Use(n,9) + 1;
% Document the node is asleep
elseif Node_Status(n,2) == 1 % Node Waking Up
            Energy_Use(n,2) = Energy_Use(n,2) + 1;
% Document the node is waking up
elseif Node_Status(n,2) == 2 % Node Waiting
            Energy_Use(n,6) = Energy_Use(n,6) + 1;
% Document the node is Waiting
elseif Node_Status(n,2) == 3 % Node Transmitting
            Energy_Use(n,4) = Energy_Use(n,4) + 1;
% Document the node is transmitting
elseif Node_Status(n,2) == 4 % Node Processing
            Energy_Use(n,3) = Energy_Use(n,3) + 1;
% Document the node is processing
elseif Node_Status(n,2) == 5 % Node Receiving
            Energy_Use(n,5) = Energy_Use(n,5) + 1;
% Document the node is receiving
elseif Node_Status(n,2) == 6 % Node Going to sleep
            Energy_Use(n,7) = Energy_Use(n,7) + 1;
% Document the node is going to sleep
elseif Node_Status(n,2) == 7 % Node Waiting Post Transmission
            Energy_Use(n,8) = Energy_Use(n,8) + 1;
% Document the node is Waiting post trans
end
```

```
end


end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Display %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Display()
% This is what initiates the events. Events are put in to create the
% sensing events that are simulated within the program.

global Events;
global Packet;
global Open_Packets;
global Timer;
global Energy_Use;
global Orig_Events;
global Initial_Payload;
global MAC_Check;
global Payload_Out;
global MAC_F;
global Routing_Table;
global fileID;
global MITMAtt;
global NodeTX;
global NodeRX;
global Packets;
global Spoofed_Node;
global Packet_Retrans_Tracker;
global Enc;
global Node_Address;


c = 0;
g = 0;
[e,f] = size(Open_Packets);
h = e-1;
j = h-Orig_Events;
z = 0;
x = 0;
for n = 2:e
if Open_Packets(n,7) == 99 %frame properly received
        c = c + 1;
elseif Open_Packets(n,7) == 98
% Transmitted 8 times unsuccessfully, placed in storage at node
        g = g + 1;
elseif Open_Packets(n,7) == 97
% Failed to be received by the BS correctly (collision)
        z = z + 1;
elseif Open_Packets(n,7) == 96
% CRC or BER on final hop to BS
```

```
        x = x + 1;

end
end
d = c/h*100;
    fprintf('\nA total of %i packets were sent and %i packets were
received for a success rate of %3.2f%%. \n\n',h,c,d);
    fprintf(fileID,'\nA total of %i packets were sent and %i packets
were received for a success rate of %3.2f%%. \n\n',h,c,d);

    fprintf('A total of %i packets were dropped within the final hop
(was not fully received by BS). \n\n',z);
    fprintf(fileID,'A total of %i packets were dropped within the final
hop (was not fully received by BS). \n\n',z);

    fprintf('A total of %i packets were dropped due to CRC or BER to
header information within the final hop.  \n\n',x);
    fprintf(fileID,'A total of %i packets were dropped due to CRC or
BER to header information within the final hop.\n\n',x);

[aaa,bbb] = size(Packet_Retrans_Tracker);
if aaa > 1
    ccc = aaa - 1;
    fprintf('%i packets were stored and retransmitted at a later time.
\n\n',ccc);
    fprintf(fileID,'%i packets were stored and retransmitted at a later
time. \n\n',ccc);
end

%variables to analyze the path indication bits
[a,b] = size(Packet);
orig = 0;
second = 0;
orignode = 0;
secnode = 0;
or = 1;
se = 1;



for n = 1:a
if Open_Packets(n,7) == 99
% checks for secondary route along follow-on nodes
if Packet(n,161) == 1
for nn = 1:26
if Node_Address(nn,50:65) == Packet(n,169:184)
                    second = 1 + second;
                    secnode(se) = nn;
                    se = se + 1;
end
end
end
% checks for secondary route at originating nodes
if Packet(n,162) == 1
for nn = 1:26
```

```
if Node_Address(nn,50:65) == Packet(n,169:184)
                    orig = orig + 1;
                    orignode(or) = nn;
                    or = or + 1;
end
end
end
end
end

% add to table for secondary routes used
for n = 1:27
    Sec_Route_Table(n,1:3) = [n 0 0];
end

[aa,bb] = size(orignode);

[cc,dd] = size(Routing_Table);

% Calculates the table to show which nodes utilized the secondary
% routesfrom the original node
if orignode > 0
for n = 1:bb
        Sec_Route_Table(orignode(n),2) = Sec_Route_Table(orignode(n),2)
+ 1;
end
end
%Calculates the table to show which nodes utilized the secondary routes
% from the original nodes along the follow-on hops
[aa,bb] = size(secnode);
next_node = 0;
if secnode > 0
for n = 1:bb
        Sec_Route_Table(secnode(n),3) = Sec_Route_Table(secnode(n),3) +
1;
end
end

    fprintf('Table: The number of packets each node sent that took a
secondary route to the Base Station.\n');
    fprintf(fileID,'Table: The number of packets each node sent that
took a secondary route to the Base Station.\n');
    fprintf('Node \t Unsuccessful First Hop \t Unsuccessful Follow-On
Hops \n');
    fprintf(fileID,'Node \t Unsuccessful First Hop \t Unsuccessful
Follow-On Hops \n');
for n = 1:26
       fprintf(' %2i \t\t %3i \t\t\t\t %3i\n', Sec_Route_Table(n,1),
Sec_Route_Table(n,2), Sec_Route_Table(n,3));
       fprintf(fileID,' %2i \t\t %3i \t\t\t\t %3i\n',
Sec_Route_Table(n,1), Sec_Route_Table(n,2), Sec_Route_Table(n,3));
end
    fprintf('\n');
    fprintf(fileID,'\n');
```

```
aaa = 0;
bbb = 0;

% Calculates the number of authenticated frames received by the MS
Packet_Auth = 0;
pa = 1;
for ch = 2:e
if Open_Packets(ch,7) == 99
        bbb = bbb + 1;
        Encryption_Decryption(ch,2);
if MAC_F(ch,1:128) == MAC_Check(ch,1:128)
if Payload_Out(ch,1:128) == Initial_Payload
                aaa = aaa + 1;
else
                fprintf('\nPacket %i Failed Payload\n ',ch);
                fprintf(fileID,'\nPacket %i Failed Payload\n ',ch);
end
else
            Packet_Auth(pa) = Open_Packets(ch,1);
            pa = pa + 1;
end
end
end
fprintf('Out of the %i packets received, %i packets matched the
original packet sent for a %3.2f%% success rate.
\n\n',bbb,aaa,(aaa/bbb*100));
fprintf(fileID,'Out of the %i packets received, %i packets matched the
original packet sent for a %3.2f%% success rate.
\n\n',bbb,aaa,(aaa/bbb*100));
[aa,bb] = size(Packet_Auth);
[dd,ee] = size(MITMAtt);
ee = ee - 1;
% Calculates the number of spoofed frames injected into the WSN
if ee > 0 || Spoofed_Node < 27
    fprintf('A MITM or Spoofing attack was conducted and %3i packets
were not Authenticated.\n\n',bb);
    fprintf(fileID,'A MITM or Spoofing attack was conducted and %3i
packets were not Authenticated.\n\n',bb);
    fprintf('The following packets failed Authentication\n');
    fprintf(fileID,'The following packets failed Authentication\n');

    cc = 1;
for n = 1:bb
        fprintf('%5i \t',Packet_Auth(n));
        fprintf(fileID,'%5i \t',Packet_Auth(n));
if cc == 10
            fprintf('\n');
            fprintf(fileID,'\n');
            cc = 0;
end
        cc = cc + 1;
end
    fprintf('\n\n');
    fprintf(fileID,'\n\n');
```

```
%creates table giving the number of unauthenticated frames from each
%originating node
    Node_Auth = zeros(1,26);
    fprintf('The following nodes had packets that did not
Authenticate\n');
    fprintf(fileID,'The following nodes had packets that did not
Authenticate\n');
    fprintf('Node \t\t # Not Authenticated\n');
    fprintf(fileID,'Node \t\t # Not Authenticated\n');
for n = 1:bb
        num = Packet_Auth(n);
        Node_Auth(Packets(num,2)) = Node_Auth(Packets(num,2)) + 1;
end
for n = 1:26
        fprintf('%i \t\t %i \n',n,Node_Auth(n));
        fprintf(fileID,'%i \t\t %i \n',n,Node_Auth(n));
end
    fprintf('\n\n');
    fprintf(fileID,'\n\n');
end

fprintf('This simulation ran for a total of %i milliseconds or %5.2f
seconds.\n',Timer, (Timer/1000));
fprintf(fileID,'This simulation ran for a total of %i milliseconds or
%5.2f seconds.\n',Timer, (Timer/1000));
fprintf('Time each node spent in each phase in milliseconds \n Node\t
WU\t Proc\t CSMA\t Tran\t   Rec\t\t RX-TX\t Wait\t GTSlp\t PTran\t
Slp\n');
fprintf(fileID,'Time each node spent in each phase in milliseconds \n
Node\t   WU\t Proc\t CSMA\t Tran\t\t   Rec\t RX-TX\t Wait\t GTSlp\t
PTran\t Slp\n');
total = zeros(1,26);
totalms = zeros(1,26);
%Creates table to display the energy use by each node
for n = 1:26
    node = n;
    wake = Energy_Use(n,2);
    proc = Energy_Use(n,3) - (Enc(n) * 4);
    trans = Energy_Use(n,4)-(NodeTX(n,2)*3);
    rec = Energy_Use(n,5) - NodeRX(n,2)*3;
    wait = Energy_Use(n,6) + NodeRX(n,2)*3;
    gtslp = Energy_Use(n,7);
    ptran = Energy_Use(n,8);
    slp =Energy_Use(n,9);
    csma = NodeTX(n,2)*2;
    rxtx = NodeTX(n,2);
    total(n) = wake * 20 + proc * 24 + csma*72 + rxtx*54 + trans * 90 +
rec * 72+ gtslp * 20 + wait * 72 + ptran * 72 + Enc(n) * 7.47 * 4 /
100;
    totalms(n) = total(n)/Timer;
    fprintf('%5.0f \t %5.0f \t %5.0f \t %5.0f \t %5.0f \t %6.0f \t
%5.0f \t %5.0f \t %5.0f \t %5.0f \t %6.0f\n',
node,wake,proc,csma,trans,rec,rxtx,wait,gtslp,ptran,slp);
```

```
    fprintf(fileID,'%5.0f \t %5.0f \t %5.0f \t %5.0f \t %6.0f \t %5.0f
\t %5.0f \t %5.0f \t %5.0f \t %5.0f \t %6.0f\n',
node,wake,proc,csma,trans,rec,rxtx,wait,gtslp,ptran,slp);

end

fprintf('\nEnergy used by each node \n Node# \t\t Total mW * ms \t\t
Total mW/ms \n');
fprintf(fileID,'\nEnergy used by each node \n Node# \t\t Total mW * ms
\t\t Total mW/ms \n');
for n = 1:26
    fprintf('%i \t\t %10.0f \t\t %3.2f\n',n,total(n),totalms(n));
    fprintf(fileID,'%i \t\t %10.0f \t\t
%3.2f\n',n,total(n),totalms(n));
end



end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% aes %
% Obtained/Adapted from: %
%
http://radio.feld.cvut.cz/personal/matejka/wiki/doku.php?id=root:en:pro
jects %
% Stepan Matejka, 2011, matejka[at]feld.cvut.cz %
% $Revision: 1.1.0 $  $Date: 2011/11/20 $  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [output] = aes(s, oper, mode, input, iv, sbit)
% AES Encrypt/decrypt array of bytes by AES.
% output = aes(s, oper, mode, input, iv, sbit)
% Encrypt/decrypt array of bytes by AES-128, AES-192, AES-256.
% All NIST SP800-38A cipher modes supported (e.g. ECB, CBC, OFB, CFB,
CTR).
% Usage example:    out = aesdecrypt(s, 'dec', 'ecb', data)
% s:                AES structure (generated by aesinit)
% oper:             operation:
%                   'e', 'enc', 'encrypt', 'E',... = encrypt
%                   'd', 'dec', 'decrypt', 'D',... = decrypt
% mode:             operation mode
%                   'ecb' = Electronic Codebook Mode
%                   'cbc' = Cipher Block Chaining Mode
%                   'cfb' = Cipher Feedback Mode
%                   'ofb' = Output Feedback Mode
%                   'ctr' = Counter Mode
%                   For counter mode you need external
AES_GET_COUNTER()
%                   counter function.
% input:            plaintext/ciphertext byte-vector with length
%                   multiple of 16
% iv:               initialize vector - some modes need it
%                   ending initialize vector is stored in s.iv, so you
%                   can use aes() repetitively to encode/decode
%                   large vector:
%                   out = aes(s, 'enc', 'cbc', input1, iv);
%                   out = [out aes(s, 'enc', 'cbc', input1, s.iv)];
%                   ...
% sbit:             bit-width parameter for CFB mode
% output:           ciphertext/plaintext byte-vector
%
% See
% Morris Dworkin, Recommendation for Block Cipher Modes of Operation
% Methods and Techniques
% NIST Special Publication 800-38A, 2001 Edition
% for details.

% Stepan Matejka, 2011, matejka[at]feld.cvut.cz
% $Revision: 1.1.0 $  $Date: 2011/10/12 $

error(nargchk(4, 6, nargin));

validateattributes(s, {'struct'}, {});
validateattributes(oper, {'char'}, {});
validateattributes(mode, {'char'}, {});
validateattributes(input, {'numeric'}, {'real', 'vector', '>=', 0, '<',
256});
if (nargin >= 5)
    validateattributes(iv, {'numeric'}, {'real', 'vector', '>=', 0,
'<', 256});
if (length(iv) ~= 16)
        error('Length of ''iv'' must be 16.');
end
end
if (nargin >= 6)
    validateattributes(sbit, {'numeric'}, {'real', 'scalar', '>=', 1,
'<=', 128});
end

if (mod(length(input), 16))
    error('Length of ''input'' must be multiple of 16.');
end

switch lower(oper)
case {'encrypt', 'enc', 'e'}
        oper = 0;
case {'decrypt', 'dec', 'd'}
        oper = 1;
otherwise
        error('Bad ''oper'' parameter.');
end

blocks = length(input)/16;
input = input(:);
```

```
switch lower(mode)

case {'ecb'}
% Electronic Codebook Mode
% ------------------------
        output = zeros(1,length(input));
        idx = 1:16;
for i = 1:blocks
if (oper)
% decrypt
                output(idx) = aesdecrypt(s,input(idx));
else
% encrypt
                output(idx) = aesencrypt(s,input(idx));
end
            idx = idx + 16;
end

case {'cbc'}
% Cipher Block Chaining Mode
% --------------------------
if (nargin < 5)
            error('Missing initialization vector ''iv''.');
end
        output = zeros(1,length(input));
        ob = iv;
        idx = 1:16;
for i = 1:blocks
if (oper)
% decrypt
                in = input(idx);
                output(idx) = bitxor(ob(:), aesdecrypt(s,in)');
                ob = in;
else
% encrypt
                ob = bitxor(ob(:), input(idx));
                ob = aesencrypt(s, ob);
                output(idx) = ob;
end
            idx = idx + 16;
end
% store iv for block passing
        s.iv = ob;

case {'cfb'}
% Cipher Feedback Mode
% --------------------
% Special mode with bit manipulations
% sbit = 1..128
if (nargin < 6)
            error('Missing ''sbit'' parameter.');
end
% get number of bits
        bitlen = 8*length(input);
% loop counter
```

```
        rounds = round(bitlen/sbit);
% check
if (rem(bitlen, sbit))
            error('Message length in bits is not multiple of
''sbit''.');
end
% convert input to bitstream
        inputb = reshape(de2bi(input,8,2,'left-msb')',1,bitlen);
% preset init. vector
        ib = iv;
        ibb = reshape(de2bi(ib,8,2,'left-msb')',1,128);
% preset output binary stream
        outputb = zeros(size(inputb));
for i = 1:rounds
            iba = aesencrypt(s, ib);
% convert to bit, MSB first
            ibab = reshape(de2bi(iba,8,2,'left-msb')',1,128);
% strip only sbit MSB bits
% this goes to xor
            ibab = ibab(1:sbit);
% strip bits from input
            inpb = inputb((i - 1)*sbit + (1:sbit));
% make xor
            outb = bitxor(ibab, inpb);
% write to output
            outputb((i - 1)*sbit + (1:sbit)) = outb;
if (oper)
% decrypt
% prepare new iv - bit shift
                ibb = [ibb((1 + sbit):end) inpb];
else
% encrypt
% prepare new iv - bit shift
                ibb = [ibb((1 + sbit):end) outb];
end
% back to byte ary
            ib = bi2de(vec2mat(ibb,8),'left-msb');
% loop
end
        output = bi2de(vec2mat(outputb,8),'left-msb');
% store iv for block passing
        s.iv = ib;

case {'ofb'}
% Output Feedback Mode
% --------------------
if (nargin < 5)
            error('Missing initialization vector ''iv''.');
end
        output = zeros(1,length(input));
        ib = iv;
        idx = 1:16;
for i = 1:blocks
% encrypt, decrypt
            ib = aesencrypt(s, ib);
            output(idx) = bitxor(ib(:), input(idx));
            idx = idx + 16;
end
% store iv for block passing
        s.iv = ib;

case {'ctr'}
% Counter Mode
% ------------
if (nargin < 5)
            iv = 1;
end
        output = zeros(1,length(input));
        idx = 1:16;
for i = (iv):(iv + blocks - 1)
            ib = AES_GET_COUNTER(i);
            ib = aesencrypt(s, ib);
            output(idx) = bitxor(ib(:), input(idx));
            idx = idx + 16;
end
        s.iv = iv + blocks;

otherwise
        error('Bad ''oper'' parameter.');
end

% ----------------------------------------------------------------------
---
% end of file



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% AES_GET_COUNTER %
% Obtained/Adapted from: %
%
http://radio.feld.cvut.cz/personal/matejka/wiki/doku.php?id=root:en:pro
jects %
% Stepan Matejka, 2011, matejka[at]feld.cvut.cz %
% $Revision: 1.1.0 $  $Date: 2011/11/20 $  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [out] = AES_GET_COUNTER(i)
% AES_GET_COUNTER Generates counter for aes.m - an example.
% Example function implemented to simulate counter for aestest.
% Change this function to correspond to your requirements.
% i:           counter call number; 1, 2, 3,...
% out:         counter value for given counter call

% Stepan Matejka, 2011, matejka[at]feld.cvut.cz
% $Revision: 1.1.0 $  $Date: 2011/10/12 $

global counter;
```

```
switch (i)
case 1
        out = counter;
case 2
        counter(15:16) = [255 0];
        out = counter;
case 3
        counter(15:16) = [255 1];
        out = counter;
case 4
        counter(15:16) = [255 2];
        out = counter;
otherwise
        error('Index out of bounds.');
end

% ----------------------------------------------------------------------
---
% end of file



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% aesdecrypt %
% Obtained/Adapted from: %
%
http://radio.feld.cvut.cz/personal/matejka/wiki/doku.php?id=root:en:pro
jects %
% Stepan Matejka, 2011, matejka[at]feld.cvut.cz %
% $Revision: 1.1.0 $  $Date: 2011/11/20 $  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [out] = aesdecrypt(s, in)
% AESDECRYPT Decrypt 16-bytes vector.
% Usage:            out = aesdecrypt(s, in)
% s:                AES structure
% in:               input 16-bytes vector (ciphertext)
% out:              output 16-bytes vector (plaintext)

% Stepan Matejka, 2011, matejka[at]feld.cvut.cz
% $Revision: 1.1.0 $  $Date: 2011/10/12 $

if (nargin ~= 2)
    error('Bad number of input arguments.');
end

validateattributes(s, {'struct'}, {});
validateattributes(in, {'numeric'}, {'real', 'vector', '>=', 0, '<',
256});
```

```
% copy input to local
% 16 -> 4 x 4
state = reshape(in, 4, 4);
% Initial round
% AddRoundKey keyexp(s.rounds*4 + (1:4))
state = bitxor(state, (s.keyexp(s.rounds*4 + (1:4), :))');
% Loop over (s.rounds - 1) rounds
for i = (s.rounds - 1):-1:1
% ShiftRows
    state = shift_rows(state, 1);
% SubBytes - lookup table
    state = s.inv_s_box(state + 1);
% AddRoundKey keyexp(i*4 + (1:4))
    state = bitxor(state, (s.keyexp((1:4) + 4*i, :))');
% MixColumns
    state = mix_columns(state, s);

end

% Final round
% ShiftRows
state = shift_rows(state, 1);
% SubBytes - lookup table
state = s.inv_s_box(state + 1);
% AddRoundKey keyexp(1:4)
state = bitxor(state, (s.keyexp(1:4, :))');

% copy local to output
% 4 x 4 -> 16
out = reshape(state, 1, 16);

% ---------------------------------------------------------------------
---
function out = mix_columns(in, s)
% Each column of the state is multiplied with a fixed polynomial
mod_pol

% Slow version
% out = zeros(size(in));
% for col = 1:4
%     for row = 1:4
%         % for each element
%         temp = 0;
%         for i = 1:4
%             % Multiplication in a finite field of
%             % row vector of poly_mat and
%             % column vector of the in
%             % finally xor
%             temp = bitxor(temp,...
%                 poly_mult(s.inv_poly_mat(row, i),...
%                 in(i, col),...
%                 s.mod_pol, s.aes_logt,s.aes_ilogt));
%         end
%         % place to out
%         out(row, col) = temp;
%     end
% end

% Faster implementation
% out = zeros(size(in));
% for col = 1:4
%     temp = poly_mult(14,in(1,col),s.mod_pol,s.aes_logt,s.aes_ilogt);
%     temp = bitxor(temp,
poly_mult(11,in(2,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     temp = bitxor(temp,
poly_mult(13,in(3,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     out(1,col) = bitxor(temp,
poly_mult(9,in(4,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     temp = poly_mult(9,in(1,col),s.mod_pol,s.aes_logt,s.aes_ilogt);
%     temp = bitxor(temp,
poly_mult(14,in(2,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     temp = bitxor(temp,
poly_mult(11,in(3,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     out(2,col) = bitxor(temp,
poly_mult(13,in(4,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     temp = poly_mult(13,in(1,col),s.mod_pol,s.aes_logt,s.aes_ilogt);
%     temp = bitxor(temp,
poly_mult(9,in(2,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     temp = bitxor(temp,
poly_mult(14,in(3,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     out(3,col) = bitxor(temp,
poly_mult(11,in(4,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     temp = poly_mult(11,in(1,col),s.mod_pol,s.aes_logt,s.aes_ilogt);
%     temp = bitxor(temp,
poly_mult(13,in(2,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     temp = bitxor(temp,
poly_mult(9,in(3,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     out(4,col) = bitxor(temp,
poly_mult(14,in(4,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
% end

% Faster faster implementation
% out = zeros(size(in));
% for col = 1:4
%     temp = s.mix_col14(in(1,col) + 1);
%     temp = bitxor(temp, s.mix_col11(in(2,col) + 1));
%     temp = bitxor(temp, s.mix_col13(in(3,col) + 1));
%     out(1,col) = bitxor(temp, s.mix_col9(in(4,col) + 1));
%     temp = s.mix_col9(in(1,col) + 1);
%     temp = bitxor(temp, s.mix_col14(in(2,col) + 1));
%     temp = bitxor(temp, s.mix_col11(in(3,col) + 1));
%     out(2,col) = bitxor(temp, s.mix_col13(in(4,col) + 1));
%     temp = s.mix_col13(in(1,col) + 1);
%     temp = bitxor(temp, s.mix_col9(in(2,col) + 1));
%     temp = bitxor(temp, s.mix_col14(in(3,col) + 1));
%     out(3,col) = bitxor(temp, s.mix_col11(in(4,col) + 1));
%     temp = s.mix_col11(in(1,col) + 1);
%     temp = bitxor(temp, s.mix_col13(in(2,col) + 1));
%     temp = bitxor(temp, s.mix_col9(in(3,col) + 1));
%     out(4,col) = bitxor(temp, s.mix_col14(in(4,col) + 1));
% end

% Faster faster faster implementation
% slice1 = zeros(4,4);
% slice2 = slice1;
% slice3 = slice1;
% slice4 = slice1;
% for col = 1:4
%     slice1(1,col) = s.mix_col14(in(1,col) + 1);
%     slice2(1,col) = s.mix_col11(in(2,col) + 1);
%     slice3(1,col) = s.mix_col13(in(3,col) + 1);
%     slice4(1,col) = s.mix_col9(in(4,col) + 1);
%     slice1(2,col) = s.mix_col9(in(1,col) + 1);
%     slice2(2,col) = s.mix_col14(in(2,col) + 1);
%     slice3(2,col) = s.mix_col11(in(3,col) + 1);
%     slice4(2,col) = s.mix_col13(in(4,col) + 1);
%     slice1(3,col) = s.mix_col13(in(1,col) + 1);
%     slice2(3,col) = s.mix_col9(in(2,col) + 1);
%     slice3(3,col) = s.mix_col14(in(3,col) + 1);
%     slice4(3,col) = s.mix_col11(in(4,col) + 1);
%     slice1(4,col) = s.mix_col11(in(1,col) + 1);
%     slice2(4,col) = s.mix_col13(in(2,col) + 1);
%     slice3(4,col) = s.mix_col9(in(3,col) + 1);
%     slice4(4,col) = s.mix_col14(in(4,col) + 1);
% end
% out = bitxor(bitxor(bitxor(slice1, slice2), slice3), slice4);

% Faster faster faster faster implementation
out = bitxor(bitxor(bitxor(...
    [s.mix_col14(in(1,1:4) + 1); s.mix_col9(in(1,1:4) + 1);
s.mix_col13(in(1,1:4) + 1); s.mix_col11(in(1,1:4) + 1)],...
    [s.mix_col11(in(2,1:4) + 1); s.mix_col14(in(2,1:4) + 1);
s.mix_col9(in(2,1:4) + 1);  s.mix_col13(in(2,1:4) + 1)]),...
    [s.mix_col13(in(3,1:4) + 1); s.mix_col11(in(3,1:4) + 1);
s.mix_col14(in(3,1:4) + 1); s.mix_col9(in(3,1:4) + 1)]),...
    [s.mix_col9(in(4,1:4) + 1);  s.mix_col13(in(4,1:4) + 1);
s.mix_col11(in(4,1:4) + 1); s.mix_col14(in(4,1:4) + 1)]);

% ---------------------------------------------------------------------
---
function p = poly_mult(a, b, mod_pol, aes_logt, aes_ilogt)
% Multiplication in a finite field

% Old slow implementation
% p = 0;
% for counter = 1:8
%     if (rem(b,2))
%         p = bitxor(p,a);
%         b = (b - 1)/2;
%     else
%         b = b/2;
%     end
%     a = 2*a;
%     if (a>255)
%         a = bitxor(a,mod_pol);
%     end
% end

% Faster implementaion
if (a && b)
    p = aes_ilogt(mod((aes_logt(a + 1) + aes_logt(b + 1)), 255) + 1);
else
    p = 0;
end

% ---------------------------------------------------------------------
---
function out = shift_rows(in, dir)
% ShiftRows cyclically shift the rows of the 4 x 4 matrix.
%
%   dir = 0 (to left)
%  | 1 2 3 4 |
%  | 2 3 4 1 |
%  | 3 4 1 2 |
%  | 4 1 2 3 |
%
%   dir ~= 0 (to right)
%  | 1 2 3 4 |
%  | 4 1 2 3 |
%  | 3 4 1 2 |
%  | 2 3 4 1 |
%

if (dir == 0)
% left
% use linear indexing in 2d array
    out = reshape(in([1 6 11 16 5 10 15 4 9 14 3 8 13 2 7 12]),4,4);
% old safe method
%     temp = reshape(in,16,1);
%     temp = temp([1 6 11 16 5 10 15 4 9 14 3 8 13 2 7 12]);
%     out = reshape(temp,4,4);
else
% right
% use linear indexing in 2d array
    out = reshape(in([1 14 11 8 5 2 15 12 9 6 3 16 13 10 7 4]),4,4);
% old safe method
%     temp = reshape(in,16,1);
%     temp = temp([1 14 11 8 5 2 15 12 9 6 3 16 13 10 7 4]);
%     out = reshape(temp,4,4);
end

% ---------------------------------------------------------------------
---
% end of file
```

%%

```
% aesencrypt %
% Obtained/Adapted from: %
%
http://radio.feld.cvut.cz/personal/matejka/wiki/doku.php?id=root:en:pro
jects %
% Stepan Matejka, 2011, matejka[at]feld.cvut.cz %
% $Revision: 1.1.0 $  $Date: 2011/11/20 $  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [out] = aesencrypt(s, in)
% AESENCRYPT  Encrypt 16-bytes vector.
% Usage:            out = aesencrypt(s, in)
% s:                AES structure
% in:               input 16-bytes vector (plaintext)
% out:              output 16-bytes vector (ciphertext)

% Stepan Matejka, 2011, matejka[at]feld.cvut.cz
% $Revision: 1.1.0 $  $Date: 2011/10/12 $

if (nargin ~= 2)
    error('Bad number of input arguments.');
end

validateattributes(s, {'struct'}, {});
validateattributes(in, {'numeric'}, {'real', 'vector', '>=', 0, '<',
256});

% copy input to local
% 16 -> 4 x 4
state = reshape(in, 4, 4);

% Initial round
% AddRoundKey keyexp(1:4)
state = bitxor(state, (s.keyexp(1:4, :))');

% Loop over (s.rounds - 1) rounds
for i = 1:(s.rounds - 1)
% SubBytes - lookup table
    state = s.s_box(state + 1);
% ShiftRows
    state = shift_rows(state, 0);
% MixColumns
    state = mix_columns(state, s);
% AddRoundKey keyexp(i*4 + (1:4))
    state = bitxor(state, (s.keyexp((1:4) + 4*i, :))');
end

% Final round
% SubBytes - lookup table
state = s.s_box(state + 1);
% ShiftRows
state = shift_rows(state, 0);
% AddRoundKey keyexp(4*s.rounds + (1:4))
state = bitxor(state, (s.keyexp(4*s.rounds + (1:4), :))');
```

```
% copy local to output
% 4 x 4 -> 16
out = reshape(state, 1, 16);

% ---------------------------------------------------------------------
---
function out = mix_columns(in, s)
% Each column of the state is multiplied with a fixed polynomial
mod_pol

% Slow version
% out = zeros(size(in));
% for col = 1:4
%     for row = 1:4
%         % for each element
%         temp = 0;
%         for i = 1:4
%             % Multiplication in a finite field of
%             % row vector of poly_mat and
%             % column vector of the in
%             % finally xor
%             temp = bitxor(temp,...
%                 poly_mult(s.poly_mat(row, i),...
%                 in(i, col),...
%                 s.mod_pol, s.aes_logt,s.aes_ilogt));
%         end
%         % place to out
%         out(row, col) = temp;
%     end
% end

% Faster implementation
% out = zeros(size(in));
% for col = 1:4
%     temp = bitxor(in(3,col),in(4,col));
%     temp = bitxor(temp,
poly_mult(2,in(1,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     out(1,col) = bitxor(temp,
poly_mult(3,in(2,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     temp = bitxor(in(1,col),in(4,col));
%     temp = bitxor(temp,
poly_mult(2,in(2,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     out(2,col) = bitxor(temp,
poly_mult(3,in(3,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     temp = bitxor(in(1,col),in(2,col));
%     temp = bitxor(temp,
poly_mult(2,in(3,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     out(3,col) = bitxor(temp,
poly_mult(3,in(4,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     temp = bitxor(in(2,col),in(3,col));
%     temp = bitxor(temp,
poly_mult(3,in(1,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
%     out(4,col) = bitxor(temp,
poly_mult(2,in(4,col),s.mod_pol,s.aes_logt,s.aes_ilogt));
```

```
% end

% Faster faster implementation
% out = zeros(size(in));
% for col = 1:4
%     temp = bitxor(in(3,col),in(4,col));
%     temp = bitxor(temp, s.mix_col2(in(1,col) + 1));
%     out(1,col) = bitxor(temp, s.mix_col3(in(2,col) + 1));
%     temp = bitxor(in(1,col),in(4,col));
%     temp = bitxor(temp, s.mix_col2(in(2,col) + 1));
%     out(2,col) = bitxor(temp, s.mix_col3(in(3,col) + 1));
%     temp = bitxor(in(1,col),in(2,col));
%     temp = bitxor(temp, s.mix_col2(in(3,col) + 1));
%     out(3,col) = bitxor(temp, s.mix_col3(in(4,col) + 1));
%     temp = bitxor(in(2,col),in(3,col));
%     temp = bitxor(temp, s.mix_col3(in(1,col) + 1));
%     out(4,col) = bitxor(temp, s.mix_col2(in(4,col) + 1));
% end

% Faster faster faster implementation
% slice1 = zeros(4,4);
% slice2 = slice1;
% slice3 = slice1;
% slice4 = slice1;
% for col = 1:4
%     slice1(1,col) = in(3,col);
%     slice2(1,col) = in(4,col);
%     slice3(1,col) = s.mix_col2(in(1,col) + 1);
%     slice4(1,col) = s.mix_col3(in(2,col) + 1);
%     slice1(2,col) = in(1,col);
%     slice2(2,col) = in(4,col);
%     slice3(2,col) = s.mix_col2(in(2,col) + 1);
%     slice4(2,col) = s.mix_col3(in(3,col) + 1);
%     slice1(3,col) = in(1,col);
%     slice2(3,col) = in(2,col);
%     slice3(3,col) = s.mix_col2(in(3,col) + 1);
%     slice4(3,col) = s.mix_col3(in(4,col) + 1);
%     slice1(4,col) = in(2,col);
%     slice2(4,col) = in(3,col);
%     slice3(4,col) = s.mix_col3(in(1,col) + 1);
%     slice4(4,col) = s.mix_col2(in(4,col) + 1);
% end
% out = bitxor(bitxor(bitxor(slice1, slice2), slice3), slice4);

% Faster faster faster faster implementation
out = bitxor(bitxor(bitxor([in(3,1:4); in(1,1:4); in(1,1:4);
in(2,1:4)],...
    [in(4,1:4); in(4,1:4); in(2,1:4); in(3,1:4)]),...
    [s.mix_col2(in(1,1:4) + 1); s.mix_col2(in(2,1:4) + 1);
s.mix_col2(in(3,1:4) + 1); s.mix_col3(in(1,1:4) + 1)]),...
    [s.mix_col3(in(2,1:4) + 1); s.mix_col3(in(3,1:4) + 1);
s.mix_col3(in(4,1:4) + 1); s.mix_col2(in(4,1:4) + 1)]);

% -------------------------------------------------------------------
---
```

```
function p = poly_mult(a, b, mod_pol, aes_logt, aes_ilogt)
% Multiplication in a finite field

% Old slow implementation
% p = 0;
% for counter = 1:8
%     if (rem(b,2))
%         p = bitxor(p,a);
%         b = (b - 1)/2;
%     else
%         b = b/2;
%     end
%     a = 2*a;
%     if (a>255)
%         a = bitxor(a,mod_pol);
%     end
% end

% Faster implementaion
if (a && b)
    p = aes_ilogt(mod((aes_logt(a + 1) + aes_logt(b + 1)), 255) + 1);
else
    p = 0;
end

% ---------------------------------------------------------------------
---
function out = shift_rows(in, dir)
% ShiftRows cyclically shift the rows of the 4 x 4 matrix.
%
%   dir = 0 (to left)
%  | 1 2 3 4 |
%  | 2 3 4 1 |
%  | 3 4 1 2 |
%  | 4 1 2 3 |
%
%   dir ~= 0 (to right)
%  | 1 2 3 4 |
%  | 4 1 2 3 |
%  | 3 4 1 2 |
%  | 2 3 4 1 |
%

if (dir == 0)
% left
% use linear indexing in 2d array
    out = reshape(in([1 6 11 16 5 10 15 4 9 14 3 8 13 2 7 12]),4,4);
% old safe method
%     temp = reshape(in,16,1);
%     temp = temp([1 6 11 16 5 10 15 4 9 14 3 8 13 2 7 12]);
%     out = reshape(temp,4,4);
else
% right
% use linear indexing in 2d array
    out = reshape(in([1 14 11 8 5 2 15 12 9 6 3 16 13 10 7 4]),4,4);
```

```
% old safe method
%     temp = reshape(in,16,1);
%     temp = temp([1 14 11 8 5 2 15 12 9 6 3 16 13 10 7 4]);
%     out = reshape(temp,4,4);
end

% ----------------------------------------------------------------------
---
% end of file
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% aesinfo %
% Obtained/Adapted from: %
%
http://radio.feld.cvut.cz/personal/matejka/wiki/doku.php?id=root:en:pro
jects %
% Stepan Matejka, 2011, matejka[at]feld.cvut.cz %
% $Revision: 1.1.0 $  $Date: 2011/11/20 $  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [output] = aesinfo(s, verb)
% AESINFO Display info about AES setting in AES structure.
% Usage:            out = aesinfo(s)
% s:                AES structure (generated by aesinit)
% verb:             verbose = 0 ... only key displayed
%                   verbose = 1 ... print all vectors/tables, words
%                                   in little-endian byte order
%                   verbose = 2 ... print all vectors/tables, words
%                                   in big-endian byte order
% output:           AES key length (128,192, or 256 bits)

% Stepan Matejka, 2011, matejka[at]feld.cvut.cz
% $Revision: 1.1.0 $  $Date: 2011/10/12 $

error(nargchk(1, 2, nargin));

validateattributes(s, {'struct'}, {});
if (nargin >= 2)
    validateattributes(verb, {'numeric'}, {'real', 'scalar', '>=', 0,
'<=', 2});
else
    verb = 0;
end

% to screen
ID = 1;

fprintf(ID, 'AES Type: AES-%d\n', s.length);
fprintf(ID, 'AES Rounds: %d\n', s.rounds);
fprintf(ID, 'GF(2^8) modulo poly: 0x%03x\n', s.mod_pol);
fprintf(ID, 'Key (%d bits/%d bytes):\n', s.length, s.bytes);
```

```
printaryh(ID, 'key8', s.key, 1, 8, 1);
if (verb)
    printaryh(ID, 'key16', s.key, 2, 8, verb);
    printaryh(ID, 'key32', s.key, 4, 4, verb);
    fprintf(ID, 'Expanded Key (%d bytes/%d words):\n', numel(s.keyexp),
numel(s.keyexp)/4);
    printaryh(ID, 'expkey8', s.keyexp', 1, 8, verb);
    printaryh(ID, 'expkey32', s.keyexp', 4, 4, verb);
    fprintf(ID, 'Log table (%d bytes):\n', numel(s.aes_logt));
    printaryh(ID, 'logt8', s.aes_logt, 1, 8, verb);
    fprintf(ID, 'Inverse log table (%d bytes):\n', numel(s.aes_ilogt));
    printaryh(ID, 'ilogt8', s.aes_ilogt, 1, 8, verb);
    fprintf(ID, 'S-Box table (%d bytes):\n', numel(s.s_box));
    printaryh(ID, 'sbox8', s.s_box, 1, 8, verb);
    fprintf(ID, 'Inverse S-Box table (%d bytes):\n',
numel(s.inv_s_box));
    printaryh(ID, 'isbox8', s.inv_s_box, 1, 8, verb);
if (0)
        fprintf(ID, 'Mix-col 2 (%d bytes):\n', numel(s.mix_col2));
        printaryh(ID, 'mixcol2', s.mix_col2, 1, 8, verb);
        fprintf(ID, 'Mix-col 3 (%d bytes):\n', numel(s.mix_col3));
        printaryh(ID, 'mixcol3', s.mix_col3, 1, 8, verb);
        fprintf(ID, 'Mix-col 9 (%d bytes):\n', numel(s.mix_col9));
        printaryh(ID, 'mixcol9', s.mix_col9, 1, 8, verb);
        fprintf(ID, 'Mix-col 11 (%d bytes):\n', numel(s.mix_col11));
        printaryh(ID, 'mixcol11', s.mix_col11, 1, 8, verb);
        fprintf(ID, 'Mix-col 13 (%d bytes):\n', numel(s.mix_col13));
        printaryh(ID, 'mixcol13', s.mix_col13, 1, 8, verb);
        fprintf(ID, 'Mix-col 14 (%d bytes):\n', numel(s.mix_col14));
        printaryh(ID, 'mixcol14', s.mix_col14, 1, 8, verb);
end
end
output = s.length;

% ---------------------------------------------------------------------
---
function printaryh(ID, name, data, pack, ionline, verb)
% print byte ary data like C ary
if (verb == 0)
return;
end
data = data(:);
% pack:
switch (pack)
case 1
        items = length(data);
        vartype = 'u8';
        format = '0x%02x';
case 2
        items = length(data)/2;
        data = reshape(data,2,items)';
if (verb == 2)
            data = fliplr(data);
end
        data = sum(data.*repmat([1 256], items, 1), 2);
        vartype = 'u16';
        format = '0x%04x';
case 4
        items = length(data)/4;
        data = reshape(data,4,items)';
if (verb == 2)
            data = fliplr(data);
end
        data = sum(data.*repmat([1 256 65536 65536*256], items, 1), 2);
        vartype = 'u32';
        format = '0x%08x';
otherwise
        error('Usupported pack parameter.')
end
fprintf(ID, '%s %s[%d] = {\n', vartype, name, items);
for i = 1:items
if (rem(i - 1, ionline) == 0)
        fprintf(ID,'        ');
end
    fprintf(ID, format, data(i));
if (i == items)
        fprintf(ID,'};\n');
break;
end
if (rem(i, ionline) == 0)
        fprintf(ID,',\n');
else
        fprintf(ID,', ');
end
end

% --------------------------------------------------------------------
---
% end of file



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% aesinit %
% Obtained/Adapted from: %
%
http://radio.feld.cvut.cz/personal/matejka/wiki/doku.php?id=root:en:pro
jects %
% Stepan Matejka, 2011, matejka[at]feld.cvut.cz %
% $Revision: 1.1.0 $  $Date: 2011/11/20 $  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function s = aesinit(key)
% AESINIT Generate structure with s-boxes, expanded key, etc.
% Usage:            s = aesinit([23 34 168 ... 39])
% key:              16 (AES-128), 24 (AES-192), and 32 (AES-256)
%                   items array with bytes of key
% s:                AES structure for AES parameters and tables
% Stepan Matejka, 2011, matejka[at]feld.cvut.cz
% $Revision: 1.1.0 $  $Date: 2011/10/12 $

validateattributes(key,...
    {'numeric'},...
    {'real', 'vector', '>=', 0, '<=', 255});

key = key(:);
lengthkey = length(key);

switch (lengthkey)
case 16
        rounds = 10;
case 24
        rounds = 12;
case 32
        rounds = 14;
otherwise
        error('Only AES-128, AES-192, and AES-256 are supported.');
end

% fill s structure
s = {};
s.key = key;
s.bytes = lengthkey;
s.length = lengthkey * 8;
s.rounds = rounds;
% irreducible polynomial for multiplication in a finite field 0x11b
% bin2dec('100011011');
s.mod_pol = 283;

% s-box method 1 (slow)
% ---------------------

%     % multiplicative inverse table
%     % first is zero, calculate rest
%     inverse = zeros(1,256);
%     for i = 2:256
%         inverse(i) = find_inverse(i - 1, s.mod_pol);
%     end
%
%     % generate s-box
%     s_box = zeros(1,256);
%     for i = 1:256
%         % affine transformation
%         s_box(i) = aff_trans(inverse(i));
%     end
%     s.s_box = s_box;
%
%     % generate inverse s-box
%     inv_s_box(s_box(1:256) + 1) = (1:256) - 1;
%     s.inv_s_box = inv_s_box;

% s-box method 2 (faster)
% -----------------------

% first build logarithm lookup table and its inverse
aes_logt = zeros(1,256);
aes_ilogt = zeros(1,256);
gen = 1;
for i = 0:255
    aes_logt(gen + 1) = i;
    aes_ilogt(i + 1) = gen;
    gen = poly_mult(gen, 3, s.mod_pol);
end
% store log tables
s.aes_logt = aes_logt;
s.aes_ilogt = aes_ilogt;
% build s-box and its inverse
s_box = zeros(1,256);
loctable = [1 2 4 8 16 32 64 128 1 2 4 8 16 32 64 128];
for i = 0:255
if (i == 0)
        inv = 0;
else
        inv = aes_ilogt(255 - aes_logt(i + 1) + 1);
end
    temp = 0;
for bi = 0:7
        temp2 = sign(bitand(inv, loctable(bi + 1)));
        temp2 = temp2 + sign(bitand(inv, loctable(bi + 4 + 1)));
        temp2 = temp2 + sign(bitand(inv, loctable(bi + 5 + 1)));
        temp2 = temp2 + sign(bitand(inv, loctable(bi + 6 + 1)));
        temp2 = temp2 + sign(bitand(inv, loctable(bi + 7 + 1)));
        temp2 = temp2 + sign(bitand(99, loctable(bi + 1)));
if (rem(temp2,2))
            temp = bitor(temp, loctable(bi + 1));
end
end
    s_box(i + 1) = temp;
end
inv_s_box(s_box(1:256) + 1) = (0:255);
% table correction (must be)
s_box(1 + 1) = 124;
inv_s_box(124 + 1) = 1;
inv_s_box(99 + 1) = 0;
s.s_box = s_box;
s.inv_s_box = inv_s_box;

% tables for fast MixColumns
mix_col2 = zeros(1,256);
mix_col3 = mix_col2;
mix_col9 = mix_col2;
mix_col11 = mix_col2;
mix_col13 = mix_col2;
mix_col14 = mix_col2;
for i = 1:256
    mix_col2(i) = poly_mult(2, i - 1, s.mod_pol);
    mix_col3(i) = poly_mult(3, i - 1, s.mod_pol);
    mix_col9(i) = poly_mult(9, i - 1, s.mod_pol);
    mix_col11(i) = poly_mult(11, i - 1, s.mod_pol);
    mix_col13(i) = poly_mult(13, i - 1, s.mod_pol);
    mix_col14(i) = poly_mult(14, i - 1, s.mod_pol);
end
s.mix_col2 = mix_col2;
s.mix_col3 = mix_col3;
s.mix_col9 = mix_col9;
s.mix_col11 = mix_col11;
s.mix_col13 = mix_col13;
s.mix_col14 = mix_col14;

% expanded key
s.keyexp = key_expansion(s.key, s.s_box, s.rounds, s.mod_pol,
s.aes_logt, s.aes_ilogt);

% poly & invpoly
s.poly_mat = [...
    2 3 1 1;...
    1 2 3 1;...
    1 1 2 3;...
    3 1 1 2];

s.inv_poly_mat =[...
    14 11 13  9;...
    9  14 11 13;...
    13  9 14 11;...
    11 13  9 14];

% end of aesinit.m
% ---------------------------------------------------------------------
---

function p = poly_mult(a, b, mod_pol)
% Multiplication in a finite field
% For loop multiplication - slower than log/ilog tables
% but must be used for log/ilog tables generation

p = 0;
for counter = 1 : 8
if (rem(b, 2))
        p = bitxor(p, a);
        b = (b - 1)/2;
else
        b = b/2;
end
    a = 2*a;
if (a > 255)
        a = bitxor(a, mod_pol);
end
end

% ---------------------------------------------------------------------
---
```

```
function inv = find_inverse(in, mod_pol)
% Multiplicative inverse for an element a of a finite field
% very bad calculate & test method
% Not used in faster version

% loop over all possible bytes
for inv = 1 : 255
% calculate polynomial multiplication and test to be 1
if (1 == poly_mult(in, inv, mod_pol))
% we find it
break
end
end
inv = 0;

% ----------------------------------------------------------------------
---
function out = aff_trans(in)
% Affine transformation over GF(2^8)
% Not used for faster s-box generation

% modulo polynomial for multiplication in a finite field
% bin2dec('100000001');
mod_pol = 257;

% multiplication polynomial
% bin2dec('00011111');
mult_pol = 31;

% addition polynomial
% bin2dec('01100011');
add_pol = 99;

% polynomial multiplication
temp = poly_mult(in, mult_pol, mod_pol);

% xor with addition polynomial
out = bitxor(temp, add_pol);

% ----------------------------------------------------------------------
---
function expkey = key_expansion(key, s_box, rounds, mod_pol, aes_logt,
aes_ilogt)
% Expansion of key

% This is old version for AES-128 (192?, 256? not tested):
% rcon = ones(1,rounds);
% for i = 2:rounds
%     rcon(i) = poly_mult(rcon(i - 1), 2, mod_pol);
% end
% % fill bytes 2, 3, and 4 by 0
% rcon = [rcon(:), zeros(rounds, 3)];
%
% kcol = length(key)/4;
% expkey = (reshape(key, kcol, 4))';
% for i = (kcol + 1):(4*rounds + 4)
%     % copy the previous row of the expanded key into a buffer
%     temp = expkey(i - 1, :);
%     % each fourth row
%     if (mod(i, 4) == 1)
%         % shift temp
%         temp = temp([2 3 4 1]);
%         % s-box transform
%         temp = s_box(temp + 1);
%         % compute the current round constant
%         r = rcon((i - 1)/4, :);
%         % xor
%         temp = bitxor(temp, r);
%     else
%         if ((kcol > 6) && (mod(i, kcol) == 0))
%             temp = s_box(temp);
%         end
%     end
%     % generate new row of the expanded key
%     expkey(i, :) = bitxor(expkey(i - 4, :), temp);
% end

% This is new faster version for all AES:
rcon = 1;
kcol = length(key)/4;
expkey = (reshape(key,4,kcol))';
% traverse for all rounds
for i = kcol:(4*(rounds + 1) - 1)
% copy the previous row of the expanded key into a buffer
    temp = expkey(i, :);
% each kcol row
if (mod(i, kcol) == 0)
% rotate word
        temp = temp([2 3 4 1]);
% s-box transform
        temp = s_box(temp + 1);
% xor
        temp(1) = bitxor(temp(1), rcon);
% new rcon
% 1. classic poly_mult
% rcon = poly_mult(rcon, 2, mod_pol);
% 2. or faster version with log/ilog tables
% note rcon is never zero here
% rcon = aes_ilogt(mod((aes_logt(rcon + 1) + aes_logt(2 + 1)), 255) +
1);
        rcon = aes_ilogt(mod((aes_logt(rcon + 1) + 25), 255) + 1);
else
if ((kcol > 6) && (mod(i, kcol) == 4))
            temp = s_box(temp + 1);
end
end
% generate new row of the expanded key
    expkey(i + 1, :) = bitxor(expkey(i - kcol + 1, :), temp);
end
```

```
% ---------------------------------------------------------------------
---
% end of file




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Encryption_Decryption %
% Obtained/Adapted from: %
%
http://radio.feld.cvut.cz/personal/matejka/wiki/doku.php?id=root:en:pro
jects %
% Stepan Matejka, 2011, matejka[at]feld.cvut.cz %
% $Revision: 1.1.0 $  $Date: 2011/11/20 $  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Encryption_Decryption(PN, Type)
%modified version of code from :

global counter;
global Packet;
global Keys;
global Node_Address;
global ct1;
global Payload;
global MAC;
global MAC_Check;
global Payload_Out;
global MAC_F;



    Payload(1,1:128) = zeros(1,128);
    MAC_F(1,1:128) =  zeros(1,128);
    MAC_Check(1,1:128) =  zeros(1,128);

    st = 297;
    sst = st + 7;
    aa = 1;
for n = 1:16
        pt(n) = bi2de(Packet(PN,st:sst));
        st = st + 8;
        sst = st + 7;
end

for nn = 1:26
if Packet(PN,169:184) == Node_Address(nn,50:65)
            cc = 1;
            ccc = 8;
for nnn = 1:16
                key(nnn) = bi2de(Keys(nn,cc:ccc));
                cc = cc + 8;
                ccc = cc + 7;
```

```
end
end
end

    s = aesinit(key);
    count = Packet(PN,169:296);
    MAC_Out = Packet(PN,865:992);
    st = 1;
    sst = st + 7;
    aa = 1;
for n = 1:16
        counter(n) = bi2de(count(st:sst));
         MAC_O(n) = bi2de(MAC_Out(st:sst));
        st = st + 8;
        sst = st + 7;
end



    ivh1 = {'00''00''00''00''00''00''00''00'...
'00''00''00''00''00''00''00''00'};
    iv1 = hex2dec(ivh1);

%____________________________________________________________________


if Type == 1
        ct1 = aes(s, 'enc', 'cbc', pt, iv1);
        pt(17:32) = ct1(1:16);

% MSG and IV go in
        ct = aes(s, 'enc', 'ctr', pt);
% Output is the cipher Text to be used as payload
        nn = 1;
        nnn = 8;
         p = 17;

for n  = 1:16
            Payload(PN,nn:nnn) = de2bi(ct(n),8);
            MAC(nn:nnn) = de2bi(ct(p),8);
            nn = nn + 8;
            nnn = nn + 7;
            p = p + 1;
end


elseif Type == 2

        pt(17:32) = MAC_O;
        outm(PN,1:16) = MAC_O;
%         outp(PN,1:16) = pt(1:16);
        ct2 = aes(s, 'dec', 'ctr', pt);
% Gives cipher text
```

```
        p = 17;
        nn = 1;
        nnn = 8;
for n  = 1:16
            Payload_Out(PN,nn:nnn) = de2bi(ct2(n),8);
            MAC_F(PN,nn:nnn) = de2bi(ct2(p),8);
            nn = nn + 8;
            nnn = nn + 7;
            p = p + 1;
end
                nn = 1;
        nnn = 8;
        check = aes(s, 'enc', 'cbc', ct2(1:16), iv1);
for n  = 1:16
            MAC_Check(PN,nn:nnn) = de2bi(check(n),8);
            nn = nn + 8;
            nnn = nn + 7;
end

end


end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% crc16 %
% Obtained/Adapted from: %
% https://macroware.wordpress.com/2011/05/21/crc-calculation/ %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [resto] = crc16(h)
%Variable initialization (CRC16 polynomial) in format
% [1 g1X g2X^2 g3X^3 g4X^4 g5X^5 g6X^6 g7X^7 g8X^8 g9X^9 g10X^10
g11X^11 g12X^12 g13X^13 g14X^14 g15X^15 1X^16]
gx=[1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1];
crcSize = 16;
% message ux (lsb first)
ux = h;
%ux = [0 1 0 0 0 0 0 0   0 0 0 0 0 0 1 0   0 0 0 0 0 0 0 1   0 0 0 0 0
0 0 0]
msgSize = 1000;
bx=[zeros(1,16)];
% shift the message through the generator polynomial to determine the
% remainder bx
for i=1:msgSize
    feedback = xor(ux(i), bx(16));
for j=crcSize:-1:2
if (gx(j) == 1)
            bx(j) = xor(bx(j-1), feedback);
else
            bx(j) = bx(j-1);
end
end
```

```
    bx(1) = feedback;
end
%convert the resulting remainder to a hex representation
remainder = uint16(0);
j = crcSize;
for i=0:crcSize-1
    remainder = remainder + bx(j) * 2^i;
    j = j - 1;
end
resto = de2bi(remainder,16);
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] K. White and P. Thulasiraman, “Energy efficient cross layer load balancing in tactical multigateway wireless sensor networks,” in *Proc. Of IEEE International Inter-Disciplinary Conference on Cognitive Method in Situation Awareness and Decision Support* , 2015, pp.193–199.

[2] P. N. Edwards, “We defend every place: Building the cold war world,” in *The Closed World: Computers and the Politics of Discourse in Cold War America.*Cambridge, MA: MIT Press, 1996, pp. 3–8.

[3] *Unattended Ground Sensor Set AN/GSQ-257 Technical Manual, TM 09632A-OI*, Washington, DC: U.S. Marine Corps, 2008.

[4] J Hui, D Culler, and S Chakrabarti, “6LoWPAN: Incorporating IEEE 802.15. 4 into the IP architecture” in *Internet Protocol for Smart Objects Alliance*, 2009.

[5] J. Granjal, E. Monteiro, and J. Sa Silva, “Security for the Internet of things: a survey of existing protocols and open research issues,”in *IEEE Communication Surveys & Tutorials*, vol. 17, pp.1294–1312, Third Quarter, 2015.

[6] Jonas Olsson. (2014) 6LoWPAN demystified. [Online]. Available: http://www.ti.com/lit/wp/swry013/swry013.pdf

[7] *Sensor Mobile Monitor System AN/MSC-77 Technical Manual, TM 09856A-10/1A*, U.S. Marine Corps, Washington, DC, 2008.

[8] D. W. Courtney and P. Thulasiraman, “Implementation of secure 6LoWPAN communications for tactical wireless sensor networks,” in *IEEE Conference on Computer Communications Workshops: IEEE Infocom MiseNet Workshop,* San Francisco, CA, 2016, pp. 962–967.

[9] A. Callanan and P. Thulasiraman, “Achieving sink node anonymity under energy constraints in tactical wireless sensor networks,” in *Proc. of IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, 2015, pp.186–192.

[10] S. Lee, S. Lee, H. Song, and H. Lee, “Wireless sensor network design for tactical military applications: remote large-scale environments,” in *Proc. of IEEE Military Communications Conference*, 2009, pp. 911–917.

[11] H. Song, S. Lee, S. Lee, and H. Lee, “6LoWPAN-Based tactical wireless sensor network architecture for remote large-scale random deployment scenarios,” in *Proc. of IEEE Military Communications Conference*, 2009, pp.1–7.

[12] P. Thulasiraman, “RPL routing for multigateway AMI networks under interference constraints” in *Proc. of IEEE International Conference on Communications – Selected Areas in Communications Symposium*, 2013, pp. 4477-4482.

[13] C. Hennebert and J. D. Santos, “Security protocols and privacy issues into 6LoWPAN stack: a synthesis,” in *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 384–398, Oct. 2014.

[14] S. Raza, S. Duquennoy, and G. Selander, “Compression of IPsec AH and ESP Headers for Constrained Environments (Draft),” pp.1–10, 2013, Sept. 2013.

[15] A. Rghioui, M. Bouhorma, and A. Benslimane, “Analytical study of security aspects in 6LoWPAN networks,” in *5th International Conference on Information and Communication Technology for the Muslim World*, 2013, pp.1–5.

[16] E. Dulaney and C. Easttom, “Mastering TCP/IP,” in *CompTIA Security+ Study Guide,* 6th ed. Indianapolis, IN: JW&S, ch . 3, pp. 77.

[17] Y. Lin, R. Hwang, and F. Baker, “Link Layer,” in *Computer Networks: An Open Source Approach*, New York, NY: MGH Co, 2012, pp. 156–157, 173–175.

[18] National Security Agency (2015, Aug.). NSA Suite B Cryptography. Suite B [Online]. Available: https://www.nsa.gov/ia/programs/suiteb_cryptography/.

[19] J. Hui and P. Thubert, Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks, Request For Comments (RFC): 6282, Sept. 2011.

[20] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen, “How low energy is Bluetooth low energy? Comparative measures with ZigBee/802.15.4,”*WCNC Workshop on Internet of Things Enabling Technologies, Embracing Machine-To-Machine Communications and Beyond*, 2012, pp. 232–237.

[21] J. Lee, K. Kapitanova, and S. H. Son, “The price of security in wireless sensor networks,” in Computer Networks, vol. 54, no. 17, pp. 2967–2978, Dec. 2010.

[22] (2011, May 21). CRC Calculation [Online]. Available: https://macroware.wordpress.com/2011/05/21/crc-calculation/

[23] S. Matejka. (2011, November, 20). AES - AES-128, AES-192, and AES-256 encryption/decryption functions [Online]. Available: http://radio.feld.cvut.cz/personal/matejka/wiki/doku.php?id=root:en:projects

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California